1993

Digital correction of dynamic nonlinearities in digital-to-analog converters

George R. Spalding Jr.
Iowa State University

Order Number 9424262

Digital correction of dynamic nonlinearities in digital-to-analog converters

Spalding, George R., Jr., Ph.D.

Iowa State University, 1994

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

Digital correction of dynamic nonlinearities in digital-to-analog converters

by

George R. Spalding, Jr

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY
Department: Electrical Engineering and Computer Engineering
Major: Electrical Engineering

Approved:

Signature was redacted for privacy.
In Charge of Major Work

Signature was redacted for privacy.
For the Major Department

Signature was redacted for privacy.
For the Graduate College

Members of the Committee:

Signature was redacted for privacy.

Iowa State University
Ames, Iowa
1993

To my Mother and Father, for all their support

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I wish to thank my major professor, Dr. Randy Geiger, for his enthusiastic instruction and thoughtful guidance, and the late Dr. Peter VanPeteghem for the direction he provided early in my studies.

Thanks to Dr. Jennifer Davidson, Dr. Marwan Hassoun, Dr. Stan Burns, and Dr. Jo Min for serving on my committee, to Tom Petersen and IBM, Rochester, MN for funding this research, and to the members of the Video Image Processing Group, Analog Devices Semiconductor, Wilmington, MA, for all their suggestions and assistance.

Thanks also to Peter Real, Analog Devices BV, Limerick, Ireland, for having patients when I really needed it, to R.C. Waits of Texas A&M University for providing valuable PC board fabrication and technical assistance, and Roberta "Bert" North for making it all run smoothly.

1. THE FREQUENCY DOMAIN IMPLICATIONS OF DAC NONIDEALITIES

1.1 Introduction

Today's communication systems and test instruments demand a signal generator that is stable, spectrally pure, and controllable in amplitude, frequency, and phase. There are both analog and digital approaches to signal generator design, and each has its own advantages.

At the heart of all analog-based signal generators is an oscillator that requires a high-Q resonant element, a feedback loop, an integrating/wave-shaping circuit, or a combination of these components [1]. Although analog oscillators can produce stable, spectrally-pure, fixed-frequency sine waves with low component count and little complexity, they have difficulty in combining these features with wide frequency tuning ranges and phase control [1, 2]. The tuning range limitation is due to the fact that the frequency is usually nonlinearly related to the value of several passive components or the resonant value of a high-Q crystal. Because of the typical nature of this nonlinear relationship, to change the frequency by more than an order of magnitude, the component in question usually must be switched with another. This can cause unwanted transients in the signal as well as other problems. Phase control can be implemented by surrounding the oscillator with a phase-locked loop (PLL) [3] or tunable filter loop [4], but fast phase transients are difficult to achieve with these structures.

Direct Digital Synthesis (DDS) is a way of generating waveforms of arbitrary shape and frequency from a single, high-frequency reference clock[2]. A phase accumulator uses the reference clock to generate the phase (and therefore the frequency) of the output, then a digital memory element is used as a table lookup to generate

the amplitude information, in digital format, from the phase. A Digital-to-Analog Converter (DAC) provides the analog signal from the digital amplitude information.

The advantages of a DDS-based system over a more conventional analog design are:

- Modulation can be built into the digital portion of the system, eliminating the need for nonlinear analog blocks.
- Wide output frequency range is attainable without loss of signal quality.
- The system can produce fast frequency and phase transients.
- Arbitrary waveforms can be created, both periodic and single event.
- No additional passive elements are needed to fix phase or frequency.

Applications for a DDS system include:

- Standard, stand-alone signal generators
- Test instruments in which nonstandard waveforms are needed (single events, nonrectangular pulses, radar chirps, etc.)
- Spread-spectrum communications [5] where the carrier frequency is also changing (mobile carrier)

The object of the work reported here is to design a digital-to-analog converter system for use in waveform generation which utilizes direct digital synthesis techniques, and yet is capable of substantially greater dynamic range than existing systems. Such a system places unique requirements on the DAC design. The result of this research achieves spectral purity as good as currently-available DAC-based systems, and the potential for even greater performance clearly exists.

Unlike conventional designs where time-domain specifications play a major role, DDS systems are specified almost completely in the frequency domain. Before entering into design considerations, it is necessary to analyze the DDS system as a whole and derive the appropriate DAC requirements from that analysis.

Important requirements of a DDS system are:

- Large spurious-free dynamic range
- High-frequency signal generation
- Wide-frequency tuning range
- Fine frequency resolution
- Built-in modulation capabilities
- System flexibility

Some of these requirements can be loosely translated into DAC specifications without the detailed analysis that will comprise the remainder of this chapter. Due to the importance of high-frequency dynamic range, DAC settling-time and resolution will be secondary to such items as system bandwidth and spurious-free dynamic range (SFDR) [6]. This last term is defined as the ratio of the amplitude of the fundamental of a generated sine wave to that of the largest unwanted frequency spike within the bandwidth of the system. Although this spike may be one of the sine wave's harmonics, it is not necessarily so. For this reason, total harmonic distortion (THD), or the sum of the power of all harmonics, is not the best indication of a DDS DAC's spectral purity. It should be mentioned however that specifications like settling-time and resolution may affect SFDR, but only indirectly, as will be shown in the next section.

1.2 DDS System Components

A generalized block diagram of a typical DDS system appears in Figure 1.1, and descriptions of the key blocks are presented here along with some relevant comments [2, 7, 8, 9, 10].

1.2.1 Phase accumulator

The phase accumulator is a digital component that integrates the input frequency over time to calculate the instantaneous phase of the desired output. This operation is actually performed digitally by adding a scaled version of the input frequency to

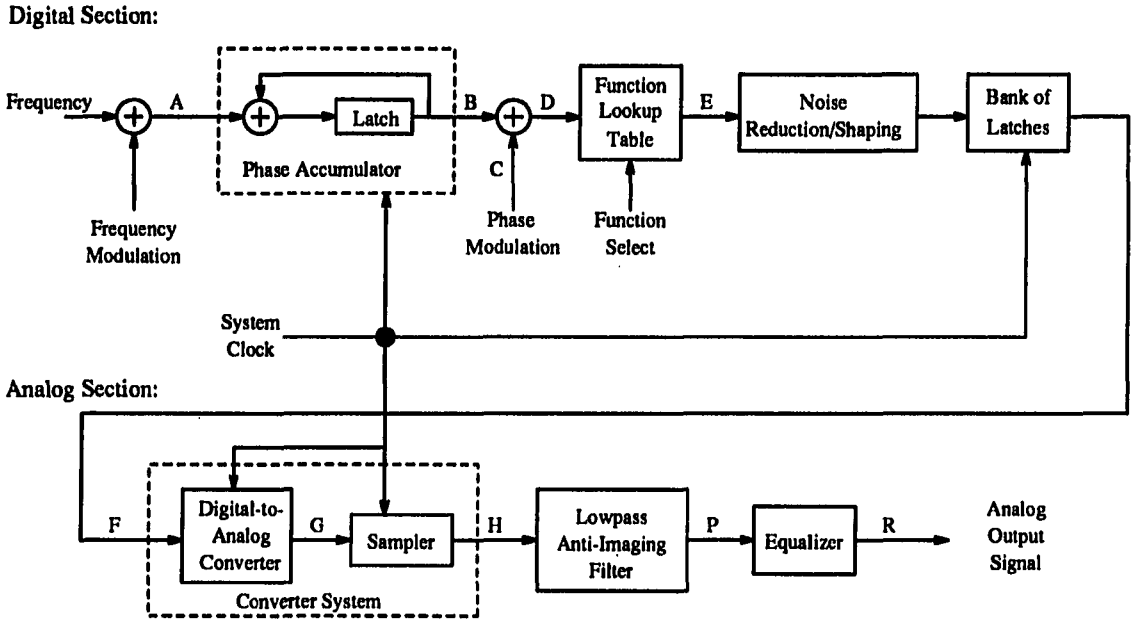

Figure 1.1: A typical configuration of a DDS signal generation system

the previously calculated phase at every clock cycle. For example, to generate a sine wave at a frequency f_o, the input to the phase accumulator, A, should be:

$$A = \frac{f_o 2^n}{f_s} \tag{1.1}$$

where f_s is the clock rate, n is the number of bits in the accumulator, and 2^n the number where roll over occurs.

1.2.2 Function lookup table

The function lookup table is a digital memory device which stores the shape of the desired output waveform. Input phase information is used as the memory address of the signal value corresponding to that phase. Normally there would be a unique memory location for each legitimate value of phase, but tricks can be played to reduce the required memory if the waveform has a particular symmetry. The advantage to this technique is that the memory is just a phase-amplitude mapping, and is completely independent of signal frequency. The width of the address (in bits)

does not have to be the same as the amplitude information.

1.2.3 Noise reduction circuitry

This digital block attempts to remove any "color" or spectral shape to the noise due to patterns in the quantization. In a coherent system, the same errors will be repeated on every period and produce spectral lines. Noise reduction will spread the energy of this distortion over a wider range of frequencies.

1.2.4 Latch bank

The latch bank is a well-timed and synchronized circuit used to remove data skew and jitter (both to be defined later) from the final digital signal.

1.2.5 Digital-to-analog converter

The DAC is the first analog block. It converts the digital input into an analog, typically piecewise-constant (staircase) signal. In most DDS systems, the DAC is the major contributor to distortion and is the limiting factor in both speed and resolution.

1.2.6 Sampler

The sampler is an optional analog block used to remove glitch-related distortion from the signal by "blocking out" any nonlinear settling of the DAC. The sampler is designed to have superior AC performance to, but maintain the DC accuracy of, the DAC. Possible sampler architectures include the return-to-zero, sample-and-hold, and track-and-hold, the properties of which will be compared later in this work.

1.2.7 Low-pass or anti-imaging filter

This analog filter is used to separate the imaged signals from the desired, base-band signal. It should look as much like a box filter with cutoff frequency of $f_S/2$ as possible. Flat response and linear phase are important in order to keep distortion down.

1.2.8 Equalizer or sinx/x compensation

The equalizer removes or compensates for sinx/x distortion introduced by the sampling function of the DAC and sampler. This is typically done with a low-pass filter which peaks in order to cancel the sinx/x droop over a limited bandwidth.

1.3 A Time Domain Example

To better understand the workings of this DDS system, consider Figure 1.2. The signals shown correspond to the node voltages in Figure 1.1. The input to the phase accumulator shown in (A) represents the frequency of the output signal which, at a certain time t_1, doubles. This input will generate the phase signal shown in (B). After time t_1, the rate of change of the phase doubles, as would be expected when the frequency doubles. The discontinuities in (B) are due to the phase accumulator "rolling over" when its maximum value is reached, just as the phase of a sine wave rolls over at 360 degrees. If the discontinuities are ignored, it is clear that (B) is the discrete-time integral of (A).

The adder between the phase accumulator and the lookup table of Figure 1.1 also has the roll over feature, allowing a time-varying signal — (C) in Figure 1.2 — to result in phase modulation.

The table lookup takes the phase signal (D) and, assuming a sine wave mapping, outputs (E). Since it is difficult to show noise reduction and sinx/x distortion/equalization in the time domain, these effects will be covered in a separate section. Note that the changes in the frequency input cause frequency modulation, while the input at point C causes phase modulation.

Figure 1.3 shows how the digital signal at the output of the table lookup is converted into an analog signal and processed to provide the desired system output signal. The digital-to-analog converter outputs a piecewise constant signal (G) that the return-to-zero sampler turns into pulses (H). The low-pass filter then removes the aliased signals, outputting a slightly distorted baseband signal. The equalizer would then compensate for this distortion, producing (R).

The phase shifts in the digital signals will vary depending on the design, but will not affect the spectrum of the final signal. In the analog blocks, however, any

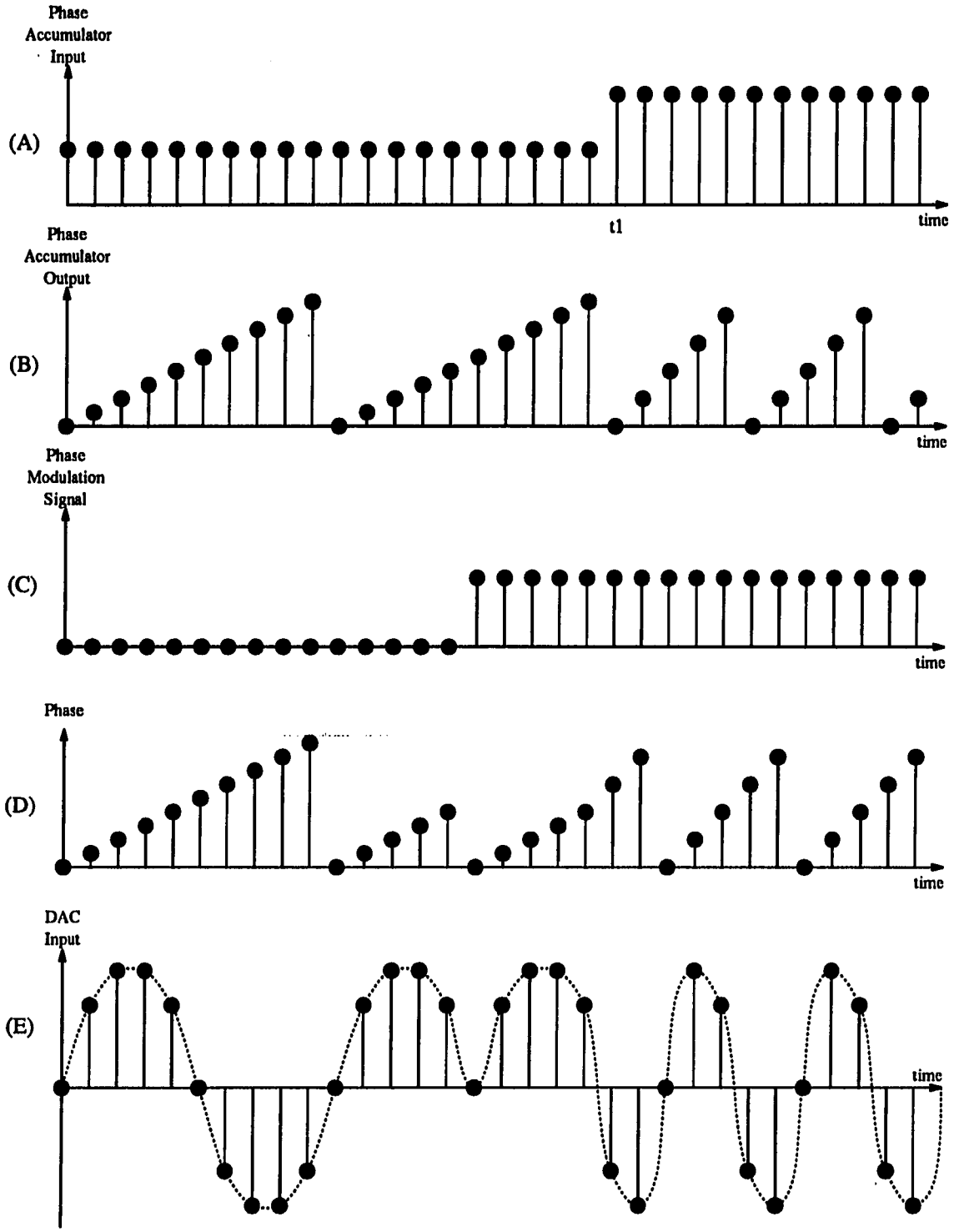

Figure 1.2: Time domain waveforms of a DDS system

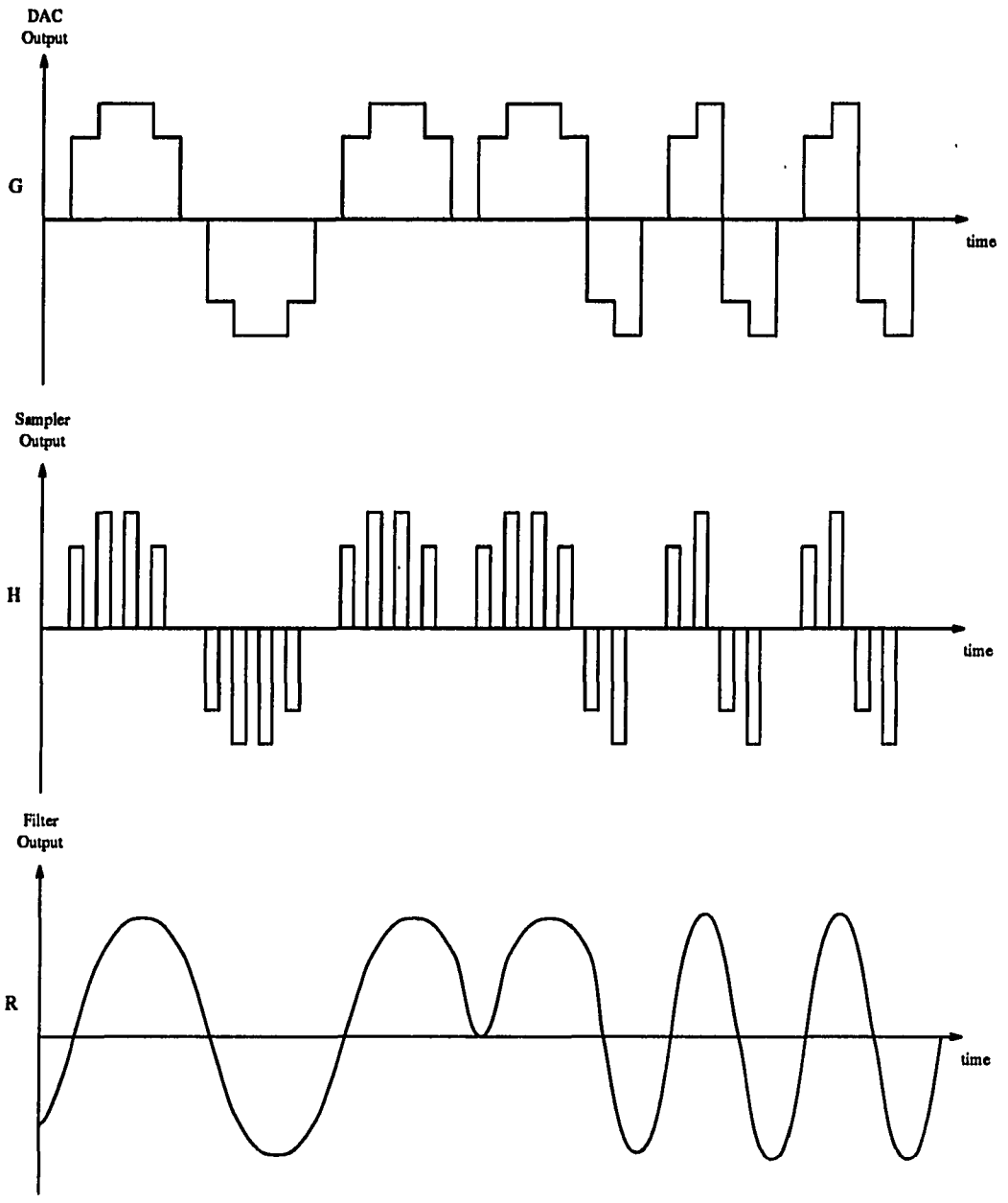

Figure 1.3: Conversion of the signal to continuous time

nonlinear phase shifts (from the filter, for example) will cause distortion.

To generate more complex or single-event signals, the inputs and the lookup table can be modified. For example, to create a one-time event like that shown in Figure 1.4 (F), the node voltages would look like those in Figures 1.4 (A-E). It should be stressed that the signals generated must be band-limited to $f_s/2$ or $\frac{1}{2T}$ to avoid extra images in the baseband signal. This limits the ability of the system to generate sharp edges and square waves. The time-domain appearance of such signals will depend heavily on the anti-imaging filter used, because some will cause high overshoot and ringing.

1.4 Memory-Based DDS Systems

An alternate method for generating the signals in the digital domain and piping them through the DAC is slightly more straightforward than has been presented thus far. A block diagram is shown in Figure 1.5. In this system, a computer calculates the digital signal and down loads it into fast memory. The computer is then disconnected, and a fast counter takes its place. Since the signal is stored in sequence, the counter simply runs through the addresses one at a time, and the output data from the memory is sent through the DAC. The reason for this configuration is that a general-purpose computer cannot output the sequence fast enough to produce very-high-frequency signals.

The advantages to this system are its simplicity and its ability to generate arbitrary signals. To generate a multi-tone signal in a phase-accumulator-based structure, either the ROM has to be changed to the proper phase mapping (limiting the tones to integer relations) or another phase accumulator must be added. In the memory-based system the signal is limited only by the amount of data storage available.

Another advantage is that there are no inherent phase errors in this scheme. In the phase accumulator, the phase position is maintained very accurately by the large accumulator, but, because of the memory needed, not all of this position is used to address the ROM. The signal is truncated or rounded to a certain phase resolution. This should not be confused with the amplitude resolution (at the output of the ROM), which is determined by the DAC. The size of the ROM is determined by the

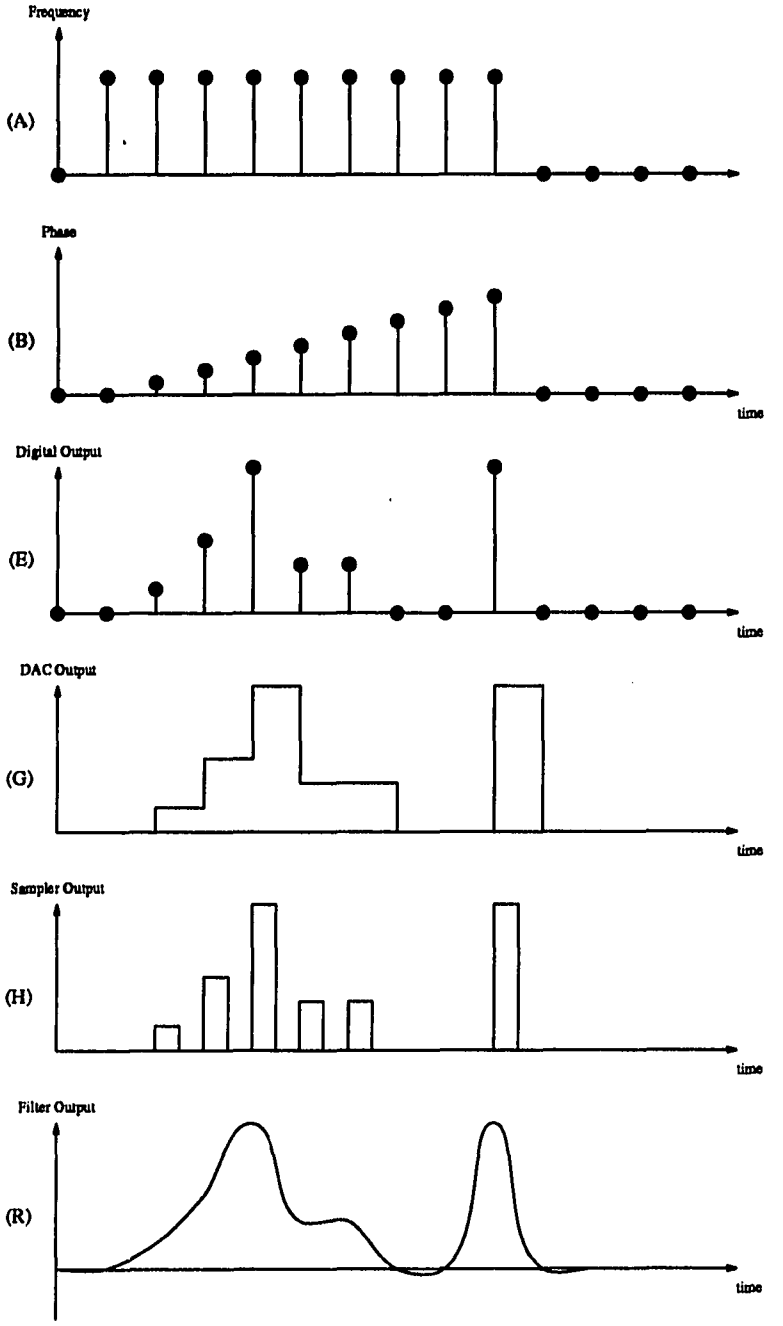

Figure 1.4: Single-shot event generation with a DDS system

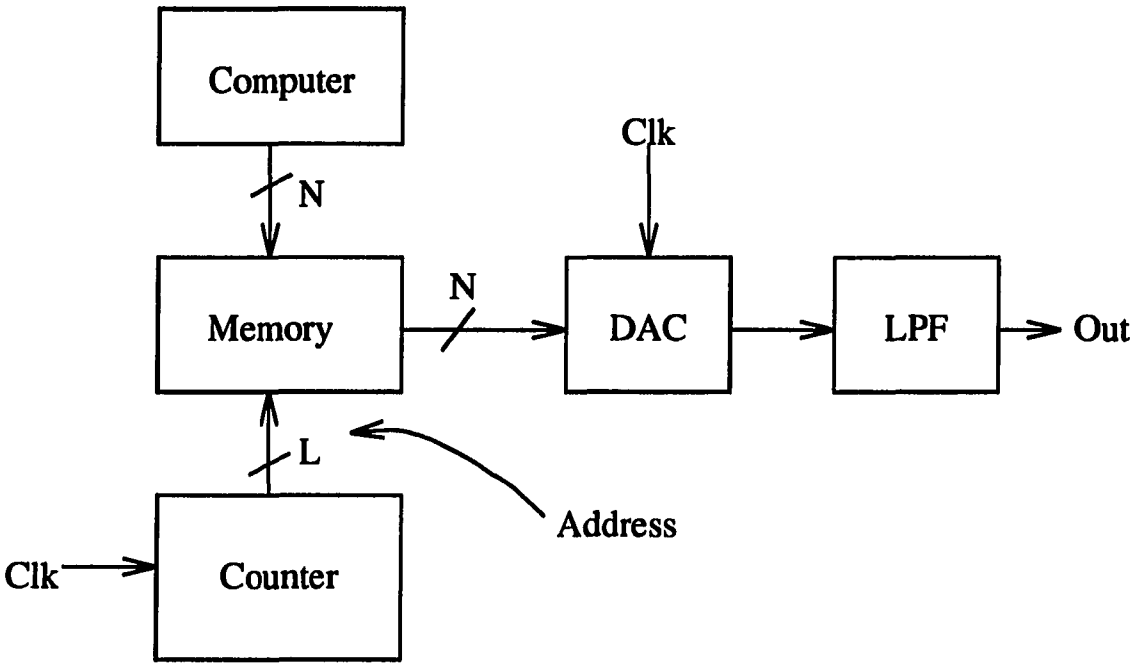

Figure 1.5: A memory-based DDS system

product of the two resolutions (address size and output word width).

The main drawback to this system is that signals cannot be generated or modulated in real time as they can be with the more classical approach. This tends to limit its applications to test systems and stand-alone signal generators.

1.5 Frequency Resolution

One of the important DDS specifications is frequency resolution, which is a measure of how closely the system can space tones in the frequency domain. In the case of the memory-based generator with a fixed clock, the length of memory determines the longest period, and therefore the lowest frequency that can be created. The next longest complete period that can be fit into memory is one half as long (and fits in twice). The frequency resolution is therefore:

$$\Delta f = f_S / L \tag{1.2}$$

where f_S is the clock rate in samples-per-second, and L is the total number of memory addresses (two raised to the number of memory address pins). In a phase accumulator the resolution is determined by the clock rate and the number of possible accumulator outputs A (two raised to the number of bits in the accumulator) as:

$$\Delta f = f_S/A \tag{1.3}$$

The implications of frequency resolution will be discussed again in the narrow-band signal generation section of Chapter 3.

1.6 Glossary

Before delving into the mathematical representation of the system, it is important to understand and define the specifications used in determining the performance of a DDS system [11, 12, 13, 14, 15, 16]. Many of the following definitions are specific to signal generation and this work, and include an indication of their relative importance for this application.

In a DDS output signal, any unwanted spectral component is termed a *frequency spur* (so called because, when displayed on a spectrum analyzer, these components look like spurs coming up out of the noise floor). *Spur-Free Dynamic Range* (SFDR) is the ratio of the signal amplitude to that of the largest spur. SFDR is one of the most important specifications in signal generation, partly because the dominant spur may not be a harmonic, reducing the effectiveness of total-harmonic distortion as an indicator of spectral performance.

The square root of the average power of a signal is called its *Root-Mean-Square* (RMS) value [15, p. 381]. For a signal $x(t)$ that is periodic with period T:

$$RMS[x(t)] = \sqrt{\frac{1}{T}\int_0^T x^2(t)dt} \tag{1.4}$$

The RMS value for a sinusoid with a peak amplitude of A is then $\frac{A}{\sqrt{2}}$. For a random signal $n(t)$, the definition is the same, but the limit must be taken as T approaches infinity. An equivalent definition derived from the mean ($E[n]$) and variance (σ^2) statistics of $n(t)$ is [16, p. 341]:

$$RMS[n(t)] = \sqrt{\sigma^2(n) + E^2(n)} \tag{1.5}$$

In sine wave generation, *Total-Harmonic Distortion* (THD) is defined as:

$$THD = \frac{\sqrt{\sum_{i=2}^{\infty} H_i}}{H_1} \tag{1.6}$$

where H_i is the RMS value of the i^{th} harmonic and H_1 is the RMS value of the fundamental.

The *Signal-to-Noise Ratio* (SNR) is the RMS signal over the RMS noise. Both device and quantization noise are summed in a root-mean sense to determine SNR.

The *Signal-to-Noise+Distortion Ratio*, is defined in the same way as SNR, but the denominator includes the RMS distortion components as well.

Dynamic range is an indication of a system's ability to process signals of various sizes. The largest signal is usually limited by the output or input range of the block, while the smallest is usually determined by the noise level, resolution, distortion, or a combination of these. Because this term has multiple definitions depending on the application, it will not be rigorously defined. Qualitatively, it will refer, in the context of this dissertation, to the ratio of the largest to the smallest signals that can be practically passed by a block.

Random *noise* from various sources will be generated in all of the analog portions of the DDS system. It is measured as an RMS quantity and specified over a particular bandwidth or as a spectral density. The significance of noise will depend a great deal on the characteristics of the output filter.

Quantization Noise (QN) is actually a deterministic signal which is easier to model as a statistical process. Finite digital resolution and accuracy result in quantization errors at the level of the least-significant bit (LSB). All digital blocks will have quantization errors associated with them because an n-bit block will output a signal at only one of 2^n discrete values.

In an ideal DAC, a one-LSB increase in the input digital code should result in a fixed increase in the analog output, regardless of the value of the code. Any deviation from the standard step size is termed *Differential NonLinearity* (DNL). If a plot of DNL vs. input code is integrated, an *Integral NonLinearity* (INL) error

curve is produced. INL is the deviation of the DAC transfer characteristic from a straight line. The equation for the line can be defined in a number of ways: converter endpoint connection, ideal predicted, or best fit (after removal of gain and DC offset errors). The maximum value of an INL or DNL plot is commonly specified in units of LSB's.

Intermodulation distortion is the distortion of a multi-tone signal that is not associated with the harmonics of the tones but rather with the intermixing of signal components (both with one another and the clock).

If a multi-tone signal is to be passed through a block without altering the relative phase of the tones, the block must have a zero or linear phase response. All other phase characteristics may lead to *phase distortion.* A linear phase block will provide a pure delay to a multi-tone signal. This will be discussed in greater detail when the effects of settling are investigated.

Settling time is the time necessary for the output of a block to reach a certain error bound around its final value after a step has been applied to its input. If the output "rings" or overshoots its final value, the time is measured until the signal enters the bound for the final time. The error bound will depend on the application, but frequently $\frac{1}{2}$LSB is used.

Update rate is the rate at which the output (usually of a DAC or ADC) is changed. In a DDS system, the clock signal to the DAC determines the update rate, and is usually the same as the digital clock rate.

For this work, *feed-through* will refer to unwanted coupling between the digital switching portions of the system and the analog signal path.

A fast transient or spike associated with a step in the output of a DAC or sampler is frequently referred to as a *glitch.* If it is code independent, it is filterable and therefore of little concern, but frequently it is a highly nonlinear function of the input code and a strong contributor to signal distortion. Glitch has been classically measured by its area, either the sum or the difference of the area above and below the ideal value. A variety of other terms such as glitch charge, glitch energy, and glitch impulse are used occasionally, each with a slightly different definition. The relative merits of these specifications and the frequency domain ramifications of glitch will be discussed later in this chapter. The circuit level origins of glitch will be covered

in Chapter 2.

The maximum rate of change of an output signal is termed *Slew Rate* (SR). It is usually caused by a fixed current charging a capacitor where the current is no longer a linear function of the input signal.

High-frequency phase noise or random variations in the period of a clock signal are known as *jitter*. Jitter on a clock signal, when viewed in the frequency domain, will appear as a spreading of the spectral lines generating "skirts".

Gain, for DDS applications, is defined as the change in analog output in response to a one-LSB change in digital input code. When it is specified as a single value, it usually refers to the slope of the line which minimizes INL errors. A *gain error* is the difference of this slope from the ideal, theoretical value. In most cases, gain errors are correctable at the output and therefore of lesser concern than the frequency domain specifications.

In the time-domain, *DC offset error* is the difference between a block's output voltage and zero when the digital code for zero is applied. This code will differ depending on the digital number system used. For frequency-domain analysis the definition changes to the difference between the intended or desired average value of the output and the actual average.

Spectral accuracy is a qualitative term that refers to how close a generated signal's spectrum is to the ideal, theoretical, or desired value. *Spectral purity*, on the other hand, refers to how free a signal is from noise and distortion. The distinction is that purity is maintained even if the signal has been scaled in amplitude or offset from its intended DC level. For DDS systems, spectral purity is the more important characteristic. The reason for this is that gain and DC offset errors can be easily removed from a spectrally pure signal after its generation, and therefore are not of particular concern, whereas a second harmonic distortion component would require signal-dependent filtering to remove.

1.7 Mathematical Representation of a Direct-Digital Synthesis System

Now that the basics of the system have been explained, the equations that describe the signals in both the time and frequency domain can be derived. There is a

one-to-one correlation between the variables used in these equations, the node voltages in Figure 1.1, and the values in Figures 1.2 and 1.6. The complete digital signal at the output of the latches in Figure 1.1 is $f(n)$, and has infinite resolution and accuracy for these ideal derivations. The Discrete-Time Fourier Transform (DTFT) of this signal, using a sample period of T, yields:

$$f(n) \overset{DTFT,T}{\longleftrightarrow} F(\omega) = \sum_{n=0}^{\infty} f(n)\exp(-jn\omega T) \tag{1.7}$$

When signal $f(n)$ is applied to the analog portion of Figure 1.1, the staircase DAC outputs:

$$g(t) = \sum_{n=0}^{\infty} f(n)[u(t-nT) - u(t-[n+1]T)] \tag{1.8}$$

and:

$$g(t) \overset{FT}{\longleftrightarrow} G(w) = \int_{-\infty}^{\infty} g(t)\exp(-jwt)dt \tag{1.9}$$

$$G(w) = \int_{-\infty}^{\infty} \sum_{n=0}^{\infty} f(n)[u(t-nT) - u(t-[n+1]T)]\exp(-jwt)dt \tag{1.10}$$

$$G(w) = \sum_{n=0}^{\infty} f(n)\left[\frac{\exp(-jwnT)}{jw} - \frac{\exp(-jw[n+1]T)}{jw}\right] \tag{1.11}$$

$$G(w) = \left[\frac{1-\exp(-jwT)}{jw}\right] \sum_{n=0}^{\infty} f(n)\exp(-jwnT) \tag{1.12}$$

$$G(w) = \left[\frac{1-\exp(-jwT)}{jw}\right] F(w) \tag{1.13}$$

The digital-to-analog conversion has altered the spectrum of the signal only by the bracketed value in Equation 1.13, which is a special case of the function $S(\omega)$, where the pulse width, τ, is equal to the clock period, T.

$$S(\omega) = \frac{1-\exp(-j\omega\tau)}{jw} \tag{1.14}$$

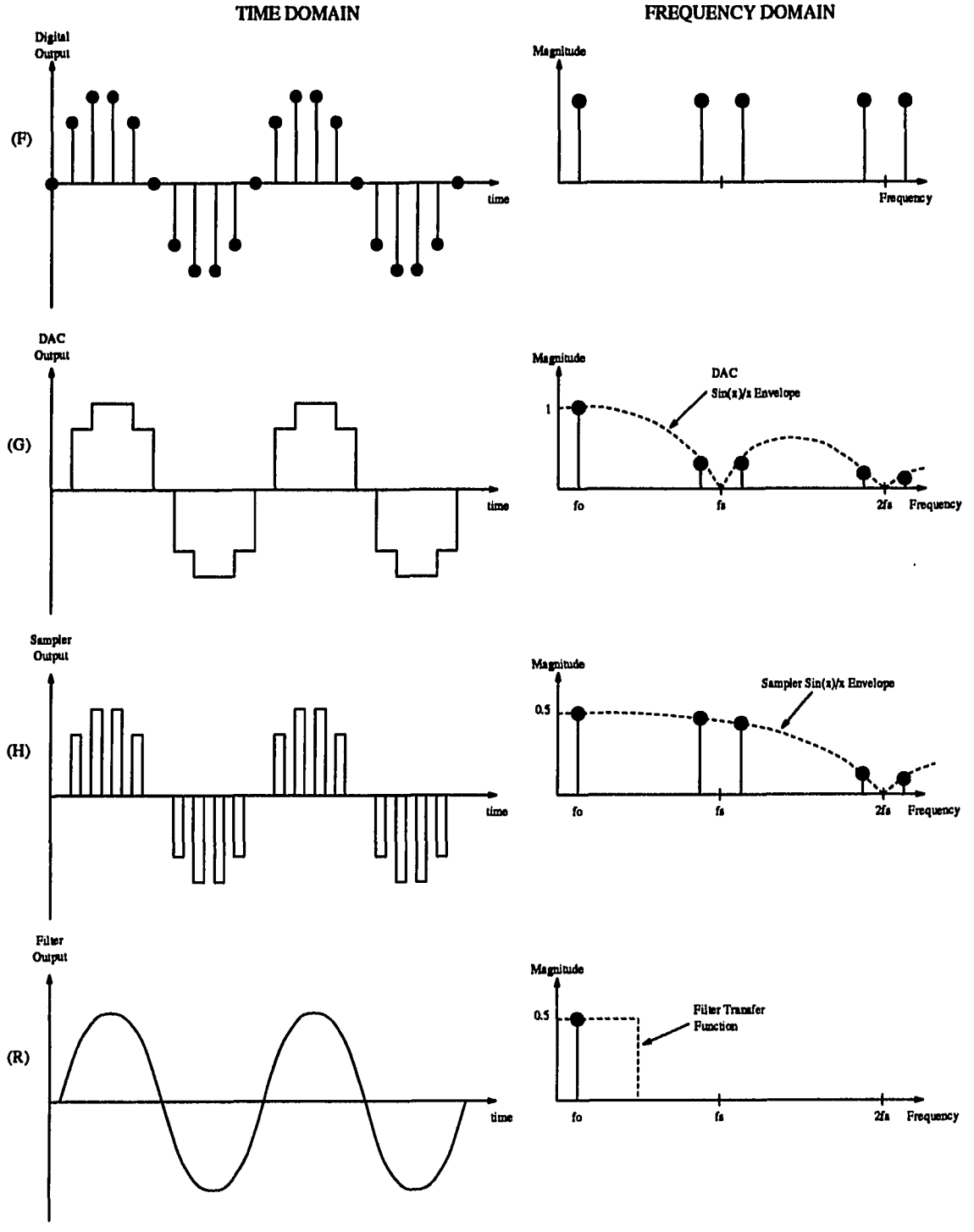

Figure 1.6: Frequency and time-domain signals of a DDS system for single-tone generation

$$S(\omega) = \left(\frac{2\exp(-j\omega\tau/2)}{\omega}\right)\left(\frac{\exp(j\omega\tau/2) - \exp(-j\omega\tau/2)}{2j}\right) \tag{1.15}$$

From Euler's Identity:

$$\sin(\theta) = \frac{\exp(j\theta) - \exp(-j\theta)}{2j} \tag{1.16}$$

Therefore:

$$S(\omega) = \frac{2\exp(-j\omega\tau/2)}{\omega}\sin(\omega\tau/2) \tag{1.17}$$

$$S(\omega) = \tau\exp(-j\omega\tau/2)\frac{\sin(\omega\tau/2)}{\omega\tau/2} \tag{1.18}$$

The exponential term is a pure delay, and τ uniformly scales the output. The remainder of the expression is known as a sinx/x envelope. If $\tau = T$, the whole expression is that of a zero-order hold [17, p. 113] . A sketch of the magnitude of $S(\omega)$ is shown in Figure 1.7 for various values of τ. One interesting case is not shown. If $S(\omega)$ is normalized by dividing by τ, and the limit is taken as τ tends to zero (as it would in an impulse train):

$$\lim_{\tau\to 0}\frac{1}{\tau}\left[\frac{1-\exp(-j\omega\tau/2)}{j\omega}\right] \approx \lim_{\tau\to 0}\frac{1}{\tau}\left[\frac{1-(1-j\omega\tau)}{j\omega}\right] = 1 \tag{1.19}$$

This implies that the spectrum will be flat with magnitude equal to the area of the sampling pulse. The linear-phase characteristic of the sampling process ensures that there will be no phase distortion. Note that narrowing the pulse width drops the amplitude of the signals passed, and moves the zeros of the transfer function higher in frequency. It may seem odd that there is not a zero at $f_S = \frac{1}{T}$ for all values of τ since the system will not pass a signal at that frequency, but it is not the sinx/x envelope which blocks that frequency, but the nature of the sampling theory and the anti-imaging filter [15, p. 435]. If a bandpass filter centered at f_S were used in place of the low-pass, and $\tau \neq T$, then a signal at f_S could be generated.

If we assume a return-to-zero sampler sampling at $\frac{T}{2}, \frac{3T}{2}, \frac{5T}{2}\ldots$ and a pulse width of γ, the output will be as in (C) of Figure 1.6 ($\gamma = \frac{T}{2}$) and can be described by:

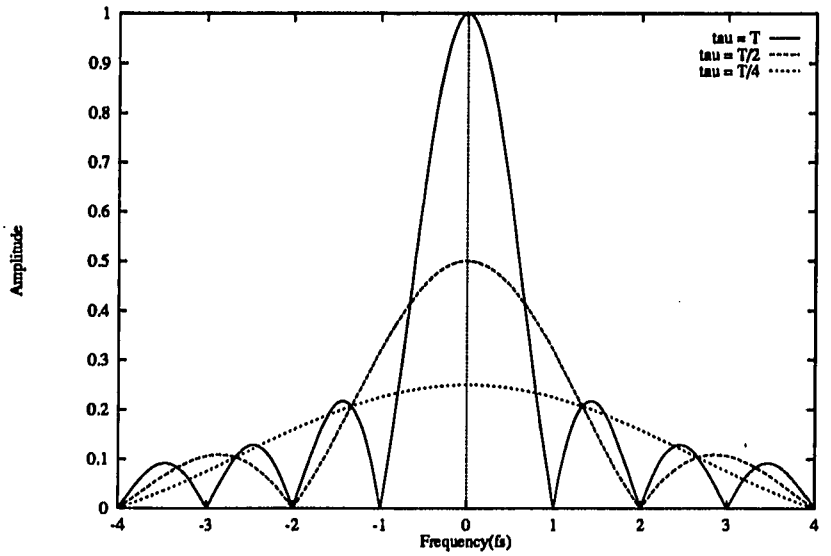

Figure 1.7: Sinx/x distortion envelope

$$h(t) = \sum_{n=0}^{\infty} f(nT)\Big[u(t - nT - \frac{T}{2}) - u(t - nT - \frac{T}{2} - \gamma)\Big] \tag{1.20}$$

$$h(t) \overset{FT}{\longleftrightarrow} H(\omega) = \exp(-jwT/2)\Big[\frac{1 - \exp(-j\omega\gamma)}{jw}\Big] \sum_{n=0}^{\infty} f(nt)\exp(-jwnT) \tag{1.21}$$

$$H(w) = \exp(-jwT/2)\Big[\frac{1 - \exp(-j\omega\gamma)}{j\omega}\Big]F(w) \tag{1.22}$$

$$H(w) = \gamma\exp(-jwT/2)\exp(-j\omega\gamma/2)\frac{\sin(\omega\gamma/2)}{\omega\gamma/2}F(w) \tag{1.23}$$

Note that this is the same signal as the output of the DAC, but with a pure delay, pulse width narrowing (due to a variation in the duty cycle), and the associated changes in the frequency domain.

An ideal brick-wall low-pass filter then chops off all signal frequencies above $f_S/2$ (called the Nyquist frequency), removing the imaged components from the desired, baseband signal. The frequency response of this filter can be described by:

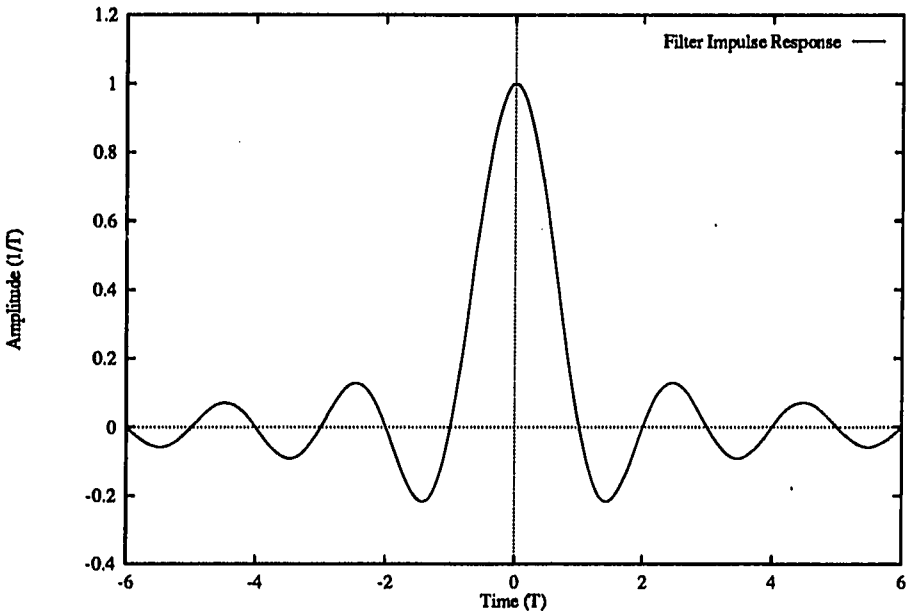

Figure 1.8: Brick-wall filter impulse response

$$X(\omega) = 1 \quad \text{for} - \frac{\pi}{T} < \omega < \frac{\pi}{T} \tag{1.24}$$

$$X(\omega) = 0 \quad \text{for all other} \quad \omega \tag{1.25}$$

yielding a time-domain impulse response of:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)\exp(j\omega t)d\omega = \frac{1}{2\pi}\int_{-\pi/T}^{\pi/T}\exp(j\omega t)d\omega \tag{1.26}$$

$$x(t) = \frac{1}{2\pi jt}\exp(j\omega t)\Big|_{-\pi/T}^{\pi/T} = \frac{1}{2\pi}\Big[\exp(j\pi t/T) - \exp(-j\pi t/T)\Big] \tag{1.27}$$

$$x(t) = \frac{1}{\pi t}\sin(\pi t/T) = \frac{1}{T}\Big[\frac{\sin(\pi t/T)}{\pi t/T}\Big] \tag{1.28}$$

as shown in Figure 1.8.

The filter outputs the signal:

$$P(\omega) = H(\omega) \quad \text{for} \quad \frac{-\pi}{T} < \omega < \frac{\pi}{T} \tag{1.29}$$

$$P(w) = 0 \quad \text{for all other} \quad \omega \tag{1.30}$$

The equalizer has a transfer function described by:

$$Y(\omega) = k\frac{\omega\gamma/2}{\sin(\omega\gamma/2)} \quad \text{for} \quad \frac{-\pi}{T} < \omega < \frac{\pi}{T} \tag{1.31}$$

$$Y(w) = ? \quad \text{for all other} \quad \omega \tag{1.32}$$

In an ideal system, the characteristics of the transfer function above $f_S/2$ are unimportant since the filter will not output any signal in that range. The output of the equalizer is the system output, and is shown in (D) of Figure 1.6. The time- and frequency-domain equations are:

$$r(t) = k\gamma f\left(t - [\frac{\gamma + T}{2}]\right) \tag{1.33}$$

$$R(\omega) = \gamma k \exp\left(-jw[\gamma + T]/2\right)F(\omega) \quad \text{for} \quad \frac{-\pi}{T} < \omega < \frac{\pi}{T} \tag{1.34}$$

$$R(\omega) = 0 \quad \text{for all other} \quad \omega \tag{1.35}$$

This shows that the analog portion of the DDS system produces an analog signal with the same time- and frequency-domain characteristics as the digital signal produced by the initial stages of the system. The only differences are in the scaling factor γk, and a pure delay of $(\gamma + T)/2$. Figure 1.6 shows the time- and frequency-domain signals for a single-tone at $f_S/10$ with $\gamma = T/2$ and $k = 1$.

1.8 Analysis of DAC Nonidealities

The mathematical description presented above will be termed the "ideal" DDS system model because the analog portion preserves the characteristics of the digital signal, and all blocks have infinite resolution, accuracy and bandwidth. In real

systems, however, the characteristics of these blocks may vary greatly from ideal. A brick wall filter, for example, is noncausal, and realizable approximations will introduce both magnitude and phase distortion. Also, because the filter will not completely block frequencies above $f_s/2$, the equalizer gain above that frequency will be of some concern. Finite digital resolution will introduce quantization errors in frequency, phase, and amplitude, and a multitude of effects in the DAC and sampler will introduce additional distortion. These various distortion components will combine to determine the overall performance of the system. This section discusses the system level cause and effect of these nonidealities, and develops mathematical models to calculate the allowable error contributions from each source.

1.8.1 Resolution, accuracy, and quantization noise

It is important to note that only a finite number of frequencies will be realizable in any DDS system because there are only a finite number of digital input signals that can be applied to the phase accumulator. Although it is possible to switch the frequency input to the phase accumulator continuously between two adjacent values in an attempt to generate an intermediate frequency, this is equivalent to frequency modulation and will not be considered a valid technique. For the remainder of this work it will be assumed that frequency resolution is fixed by the characteristics of the phase accumulator (or in the case of a memory-based system, by the size of the data storage), and is the concern of the end user.

If the digital resolution of the table lookup is smaller than that of the phase accumulator, a truncation or roundoff must be performed, resulting in phase quantization errors. If sine generation is assumed, this quantization will lead to an amplitude error of:

$$\Delta X = \sin(\phi + \Delta\phi) - \sin(\phi) \tag{1.36}$$

where ΔX is the amplitude error in fractions of full scale, $\Delta\phi$ is the error due to reduced phase resolution, and ϕ is the output of the phase accumulator. This is to be distinguished from limited phase accumulator resolution, which leads to limited frequency resolution, not to phase quantization errors. The maximum possible

amplitude error occurs when $\Delta\phi$ is at a maximum, and $\phi = 0$:

$$\Delta X_{max} = \sin(\Delta\phi_{max}) \approx \Delta\phi_{max} \tag{1.37}$$

Because of this mapping and the fact that, with sufficient word length, these errors can be made much smaller than the quantization errors of the DAC [18], only amplitude resolution and accuracy effects will be investigated in this section. The following analysis is then very similar to that in wave reconstruction systems [12, p. 541].

When generating a spectrally pure signal, precise amplitude pulses are required to keep the output error-free, but these amplitudes must be rounded off to the nearest binary code acceptable to a real-world DAC, thus introducing quantization noise (QN). If this roundoff is done to the nearest code, the QN signal will be bounded in amplitude to $\frac{1}{2}$ of the smallest step size (also called a least-significant bit, or LSB). This means that the largest possible distortion component due to quantization will have an amplitude of approximately $1/2^N$ of full scale (where N is the number of bits). For generating a single sine wave of maximum amplitude this would give a minimum signal-to-distortion ratio of 2^N or 6.02N dB.

This conservative estimate of SFDR due to QN assumes that all of the QN energy is lumped into a single frequency component. In the analysis of Analog-to-Digital Converters (ADCs) it is generally assumed that the QN is uncorrelated to the signal [19, p. 346], and can be approximated by a white noise signal with a zero-mean, rectangular probability density function as shown in Figure 1.9. Although there is obviously some correlation between the QN and the signal in a direct digital synthesis system, this approximation is still valid for determining the dynamic range limitations imposed by quantization if the steps are significantly larger than the LSB size. For an N-bit DAC with an output range of $\pm A$, the LSB size would be:

$$q = \frac{A}{2^{(N-1)}} \tag{1.38}$$

and the resulting RMS value of the zero-mean QN is:

$$RMS(q) = \sqrt{\sigma^2(q) + E^2(q)} = \sigma(q) \tag{1.39}$$

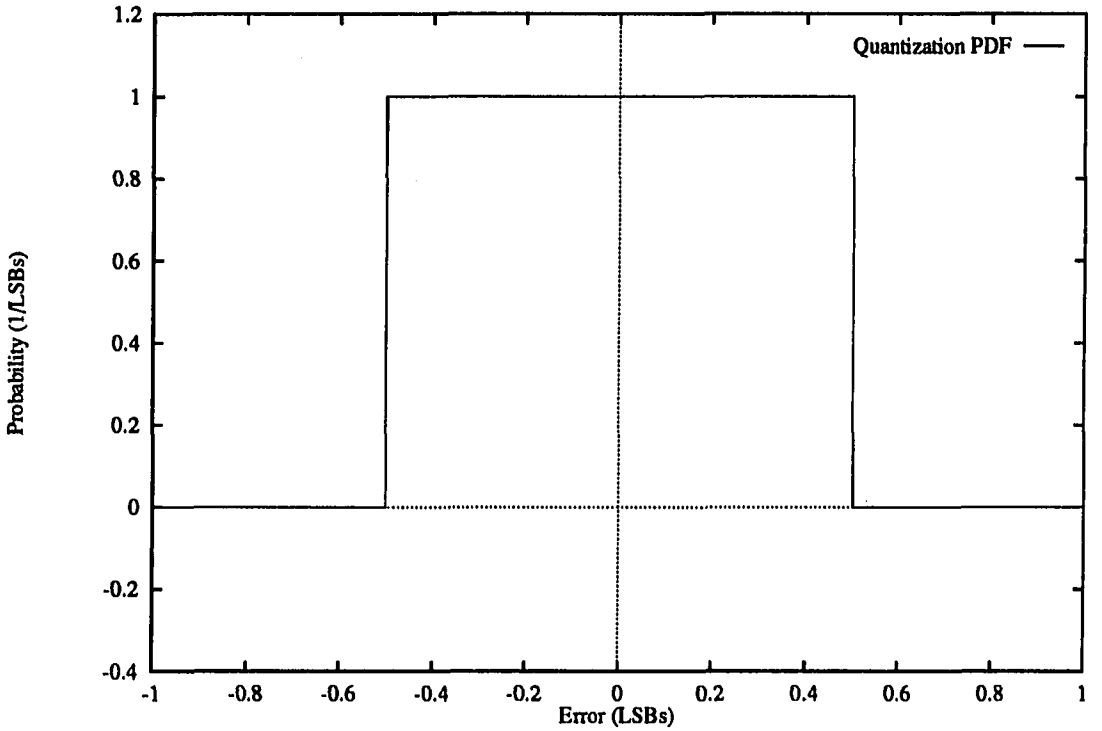

Figure 1.9: Probability density function for quantization noise

$$RMS(q) = \sqrt{\int_{-q/2}^{q/2} \frac{e^2}{q} de} = \frac{q}{\sqrt{12}} \tag{1.40}$$

The signal-to-noise ratio is therefore:

$$SN = \frac{A/\sqrt{2}}{q/\sqrt{12}} = \frac{2^{(N-1)}\sqrt{12}}{\sqrt{2}} = (1.22)2^N \tag{1.41}$$

In decibels that is:

$$SN = 20\log\Big((1.22)2^N\Big) = (6.02N + 1.76)dB \tag{1.42}$$

This dynamic range is further degraded by inaccuracies in the LSB step sizes. To better approximate the effect of quantization noise on the dynamic range of a signal, the largest DNL error should be added to the LSB size in the above equations. This is valid because the effects of DNL errors, due to their sharp discontinuities, have very rich frequency content, and thus resemble white noise.

It should be stressed that complex signals should be thought of as the sum of sinusoids. Only for periodic signals is a Fourier series representation appropriate, but finite-length signals of any description can be modeled using Fourier integral techniques. It is understood that when composing a complex signal the amplitudes of the individual components are reduced to keep the composite signal within the allowable system output range. The QN, however, will remain bounded by $\pm\frac{1}{2}$LSB.

1.8.2 Linearity

The input/output transfer characteristics of the blocks in the analog portion of a DDS system must be linear to avoid signal distortion. In many cases, device linearity degrades with increasing signal frequency, and, in the sampler and data converter, with increasing clock rate. These dynamic effects, however, are covered separately later in this chapter. For the next few paragraphs, linearity is considered a strictly DC specification.

During the analysis of converter resolution it was mentioned that DNL errors could be included in the quantization noise approximation of spectral errors. Integral nonlinearity errors, on the other hand, can be mapped directly into the frequency

domain as harmonic distortion. The ideal transfer function of a linear analog block can be written:

$$V_o = KV_i \tag{1.43}$$

where K is a constant; but in a nonlinear block:

$$V_o = f(V_i) \tag{1.44}$$

Using a Taylor series representation centered at 0 [20, p. 688], this becomes:

$$V_o = K_0 + K_1 V_i + K_2 V_i^2 + K_3 V_i^3 ... \tag{1.45}$$

$$K_n = \frac{f^n(0)}{n!} \tag{1.46}$$

To illustrate the effects of nonzero K_n ($n \neq 1$), consider the following example where $K_n = 0$ for all $n > 3$.

$$V_o = f(V_i) = K_0 + K_1 V_i + K_2 V_i^2 + K_3 V_i^3 \tag{1.47}$$

$$K_0 = f(0), K_1 = f'(0), K_2 = \frac{f''(0)}{2}, K_3 = \frac{f'''(0)}{6} \tag{1.48}$$

$$V_i = \sin(\omega t) \tag{1.49}$$

Substituting equation 1.49 into 1.47:

$$V_o = K_0 + K_1 \sin(\omega t) + K_2 \sin^2(\omega t) + K_3 \sin^3(\omega t) \tag{1.50}$$

$$V_o = K_0 + K_1 \sin(\omega t) + K_2\left[\frac{1}{2} - \frac{1}{2}\cos(2\omega t)\right] + K_3 \sin(\omega t)\left[\frac{1}{2} - \frac{1}{2}\cos(2\omega t)\right] \tag{1.51}$$

$$V_o = K_0 + \frac{K_2}{2} + K_1 \sin(\omega t) \frac{K_2}{2} \cos(2\omega t) + \frac{K_3}{2} \sin(\omega t) - \frac{K_3}{2} \sin(\omega t) \cos(2wt) \tag{1.52}$$

$$V_o = K_0 + \frac{K_2}{2} + \left[K1 + \frac{3K_3}{4}\right]\sin(\omega t) - \frac{K_2}{2}\cos(2\omega t) - \frac{K_3}{4}\sin(3\omega t) \qquad (1.53)$$

This calculation shows that:

K_0 leads to a DC offset of K_0

K_1 is the purely linear term (gain of K_1)

K_2 leads to a DC offset of $\frac{K_2}{2}$ and a second harmonic of $\frac{-K_2}{2}$

K_3 leads to a gain error of $\frac{3K_3}{4}$ and a third harmonic of $\frac{-K_3}{4}$

Figure 1.10 shows an example transfer characteristic with $K_0 = 0.1, K_1 = 1.0, K_2 = 0.1$, $K_3 = 0.1$, and the resulting INL error curve from a best-fit line.

Similar analysis will show that intermodulation distortion will result if V_i is a multi-tone signal and there are nonzero K_ns for $n > 1$. This can be quite troublesome because the products of the tone mixing can then mix with the clock and be aliased back under the cutoff frequency of the filter (if they were not less than $f_s/2$ to start with). It becomes apparent that the number of in-band frequency spurs grows rapidly with both high-order nonlinear terms and the number of tones in the signal being produced. Fortunately, the values of K_n typically fall off quickly with increasing n.

Although a DAC is inherently a nonlinear device, the above analysis still applies. This becomes clear if the DAC resolution tends to infinite, where the transfer characteristic virtually becomes a continuous function.

1.8.3 Linear settling

Ideally, the DAC and sampler should output perfect rectangular pulses. Unfortunately, resistors and capacitors at their outputs perform a filtering process that leads to settling. If this output circuitry has voltage- or current-dependent coefficients, the settling may be nonlinear. Nonlinear settling and glitch will be analyzed together since they have many of the same origins and effects.

Filtering can have two effects on a frequency-domain signal: phase distortion and amplitude distortion. To avoid amplitude distortion, the filter magnitude response must be acceptably flat over the frequencies of interest. Since the parasitic filtering of the DAC and sampler is a low-pass function like that of the anti-imaging and

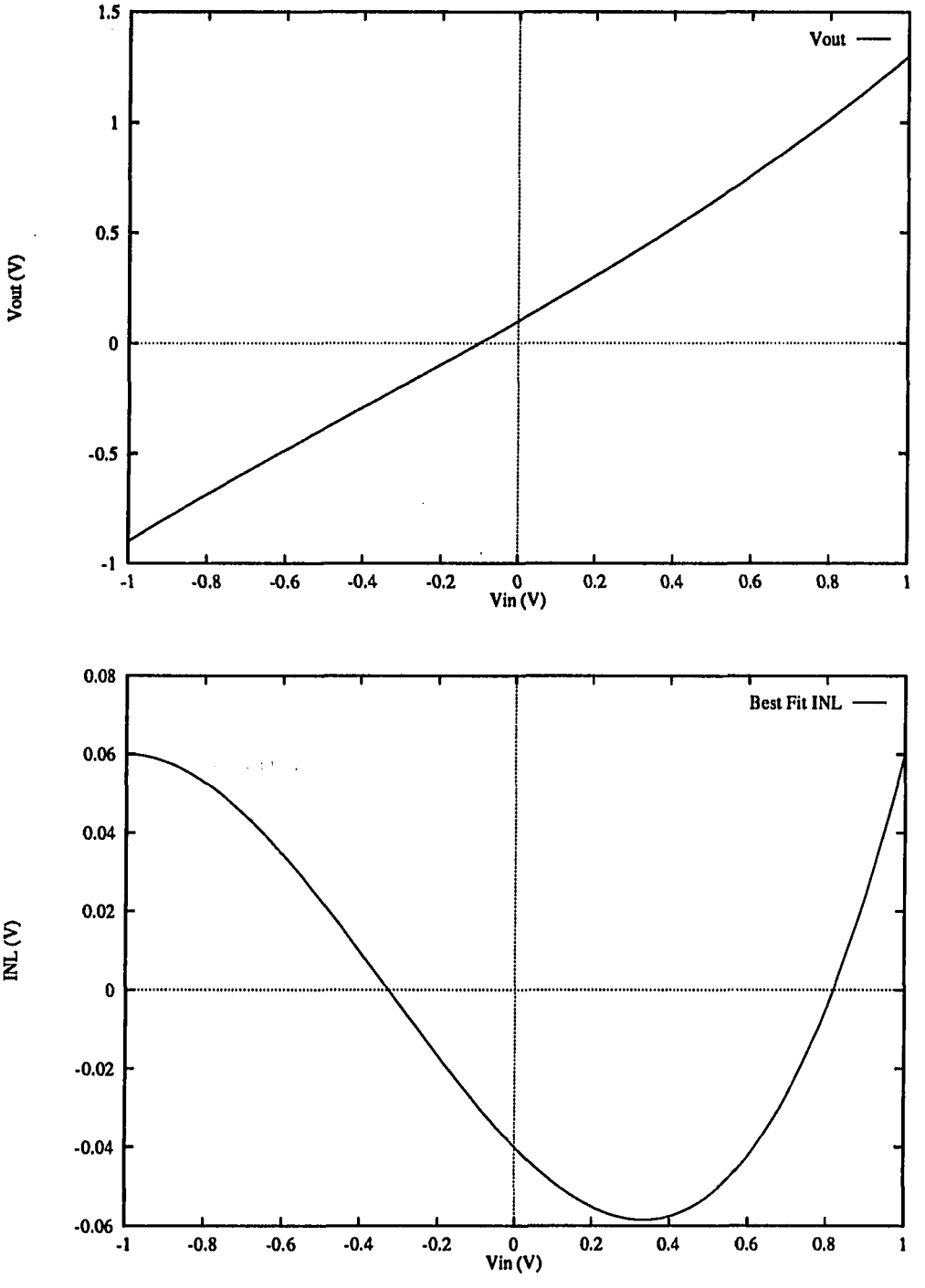

Figure 1.10: Example DC transfer characteristic (top) and the associated best-fit INL (bottom)

equalizing filters, that characteristic will be analyzed here. In general, to maintain a flat response, all of the filter poles and zeros must be sufficiently higher than the highest signal frequency. The magnitude of $H(j\omega)$ is defined as [21, p. 45]:

$$|H(j\omega)| = K\frac{\prod_{i=1}^{N}|j\omega - s_{zi}|}{\prod_{i=1}^{M}|j\omega - s_{pi}|} \tag{1.54}$$

Since most analog filters do not have a linear phase response, to avoid excessive phase distortion the poles and zeros of the filter must again be kept high to minimize the effect on the signal. The phase of a linear system at ω is defined to be [21, p. 45]:

$$\angle H(j\omega) = \sum_{i}\angle(j\omega - s_{zi}) - \sum_{i}\angle(j\omega - s_{pi}) \tag{1.55}$$

where s_{zi} are the zeros and s_{pi} are the poles.

Group delay, defined as the derivative of the phase plot with respect to frequency, is a specification used in filter design when phase distortion is an issue [22, p. 11]. A boundary placed on group delay will ensure a limited phase nonlinearity.

It was mentioned earlier that the anti-imaging or reconstruction filter cannot be a brick-wall filter since such a structure is noncausal and therefore unrealizable. In order to attenuate the aliased components of the signal, there has to exist a nonzero transition band between the highest baseband signal component and its aliased counterpart. This means that the DDS system will not be able to produce frequency components up to $f_s/2$. This "guard" band will be twice as wide as the signal bandwidth sacrificed below $f_s/2$, and will contain all of the magnitude falloff necessary to attenuate the aliased components to an acceptable level.

The design of the anti-imaging filter is a difficult task because, in addition to the magnitude and phase requirements, it must operate at very high frequencies. Many DDS systems have multiple filters that can be exchanged depending on the current application [7], and others have no filter at all because the frequency content above the Nyquist rate is of no importance. In some cases the filter used is not the product of one of the standard approximation problems like the elliptic or Chebyshev, but the result of iterative simulations and statistical design based on the signals generated by a DDS structure [23].

This work is not concerned with the filter design beyond simulation and testing needs. In many cases the signal above Nyquist will simply be ignored, but in those cases where the aliased components might mix back down due to test equipment nonlinearity (perhaps at the input to a spectrum analyzer), some high-frequency attenuation may be necessary.

It is interesting to compare time-domain and frequency-domain accuracy due to linear settling. An N-bit DAC with an effective first order output time constant of τ will settle to one half of an LSB in Δt, calculated by:

$$V_o(t) = FS\Big[1 - \exp(-t/\tau)\Big] \tag{1.56}$$

$$error = FS - V_o(t) \leq 2^{-(N+1)} FS = \frac{1}{2}\text{LSB} \tag{1.57}$$

$$\exp(-\Delta t/\tau) = 2^{-(N+1)} \tag{1.58}$$

$$\Delta t = -\tau \ln\Big[2^{-(N+1)}\Big] = \tau(N+1)\ln(2) = 0.69\tau(N+1) \tag{1.59}$$

where FS is a full-scale step. This implies that the system clock should be long enough so that Δt seconds can go by before the output is changed to avoid dropping the amplitude of the signal by more than an LSB.

Now the relationship between this time constant and the spectral purity must be determined. Since the largest deviation in the single-pole magnitude response will take place at the edge of the Nyquist rate ($\omega = \pi/T$), the frequency-domain error (or the amplitude droop) can be define as:

$$error = 1 - |X(j\omega)|_{\pi/T} \leq 2^{-(N+1)} \tag{1.60}$$

where $|X(j\omega)|$ can be approximated with the first three terms of its Taylor series expansion to yield:

$$1 - \left[1 - \frac{\pi^2\tau^2}{2T^2}\right] = 2^{-(N+1)} \tag{1.61}$$

Solving for T, the shortest period of time before another output transition, gives:

$$T = 2^{N/2}\pi\tau \tag{1.62}$$

For a 12-bit system, Equation 1.59 requires a clock period 9 times longer than τ, while Equation 1.62 dictates a factor of 201. Fortunately, precise magnitude flatness is not as important as the absence of new spectral components, such as harmonics.

The impact of these distortion types on DDS performance depends heavily on the application of the system. For single-tone generation, amplitude distortion becomes a frequency-dependent gain error, and phase distortion is a frequency-dependent delay. Even in multi-tone applications, phase distortion may be of little concern, but amplitude distortion will change the relative magnitudes of the tones. Because a linear filter does not introduce any new spectral components, it is possible to compensate for its distortion. If its characteristics are easily quantifiable, they can be removed with a digital prewarping filter (see Chapter 3).

1.8.4 Glitch and transition errors

There are numerous causes of glitch, but one of the most common and serious is a lack of synchronization in the switches of a DAC, known as data skew. Many converter structures, especially current-based DACs, are composed of binary-weighted components that are switched to ground or to the output depending on the value of the corresponding bit. If some switches are thrown earlier than others, the output will move towards an invalid output during the interim. This transient response is not linearly related to the change in digital input code, and results in distortion. Although the time constant at the output of the DAC may smooth (and thus mask) this glitch, its spectral components in the signal bandwidth will not be removed.

Consider a binary-weighted DAC constructed of bit cells that change from off-to-on a time ξ later than they make the opposite transition (which is frequently the case as will be shown in Chapter 2). This DAC's output can be decomposed into the desired signal (which still suffers from quantization, sinx/x, and other nonskew related distortion effects), and a data-skew error pulse train.

Figure 1.11 shows the output of a 3-bit DAC with this skew characteristic generating a sine wave at one sixteenth the clock rate. Figure 1.12 shows the two decom-

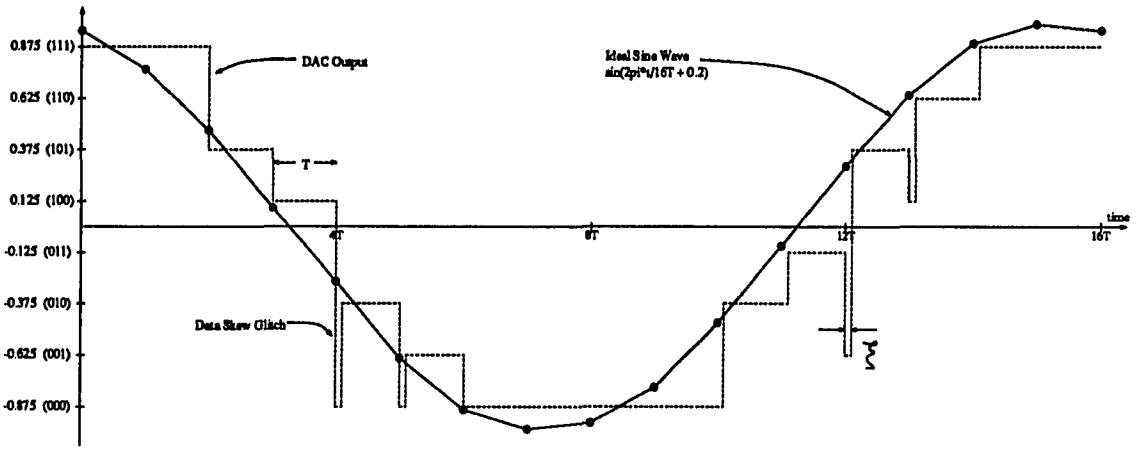

Figure 1.11: Three-bit DAC output with ongoing transition delays

position signals (with ξ equal to one tenth the clock period), while their spectrums appear in Figure 1.13. Note that the SFDR is limited by the quantization noise, and not the glitch. This is because of the low resolution of the DAC. The first skew errors appear at about -32dB, while the quantization distortion occurs in the mid-to-high 20dB range.

Although specific examples can be analyzed, it is difficult to generalize these findings into a converter dynamic-range specification because it is hard to form an analytical representation of data skew. The error signal depends nonlinearly on DC offset, number of zero crossings, and oversampling ratio, so mathematical models are difficult to develop.

This naturally leads to an attempt to employ a statistical model similar to that used on quantization errors. Unfortunately that derivation assumes virtually no correlation between the signal and the error, which is not true with skew errors unless all code transitions are equally likely, which is seldom the case.

It is important, however, to understand the relationship between the signal and skew in a qualitative sense. Assume each error pulse has a width determined by the time constants of the circuitry and process, and is, to first order, independent of the signal (this is in fact the case with most DACs being designed for spectral purposes). From the derivation of sinx/x distortion we know that this will give ξ (or more generally ξ/T), an approximately linear relation to the spectral error magnitude.

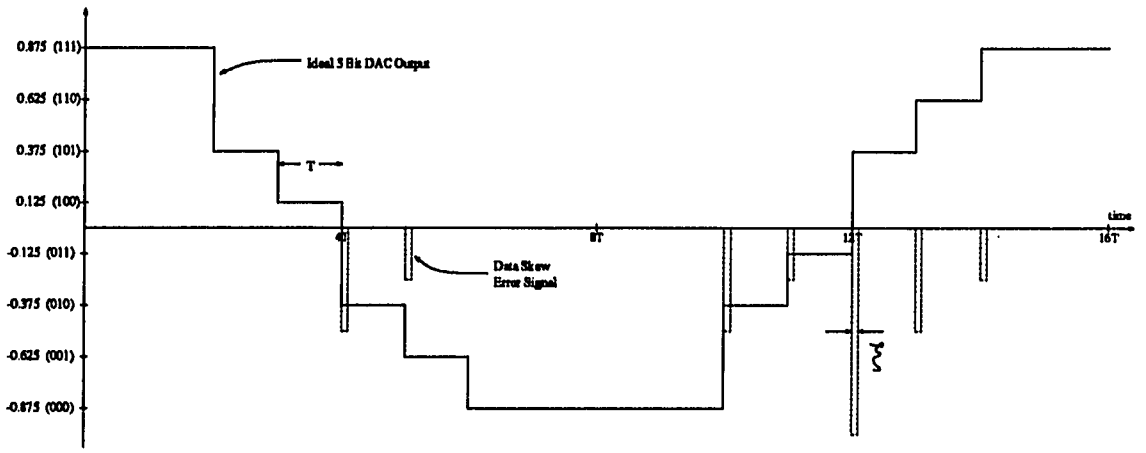

Figure 1.12: Decomposition of a signal into error and desired pulse trains

The pulse amplitude is a function of the transition's origin and destination codes. When the skew is due to a delay in the off-to-on transition, the larger the ongoing components, the larger the skew error. Note that this is *not* simply a function of the transition size, since a one-LSB change is frequently made by the switching of multiple, larger components whose difference is one LSB.

Table 1.1 shows the skew amplitudes (in fractions of full scale) of 3-bit DAC transitions versus origin and destination code. If the delay was in the offgoing transition, the signs would be positive and the matrix spun on its $x = y$ axis. These values were derived by performing a logical bitwise AND on the inverse code of origin and the positive destination code (destination & $\overline{\text{origin}}$), and then normalizing by dividing by full scale. These are the error amplitudes that would result from a binary weighted DAC, with a fixed relationship between the rise and fall of the switch signals. Here is an 8-bit example:

code	binary	decimal	hex
origin:	01101001	105	69
destination:	10110010	178	B2
error (&):	10010010	146	92

The amplitude of this error is 146/256 times the full scale range. An N-bit DAC has

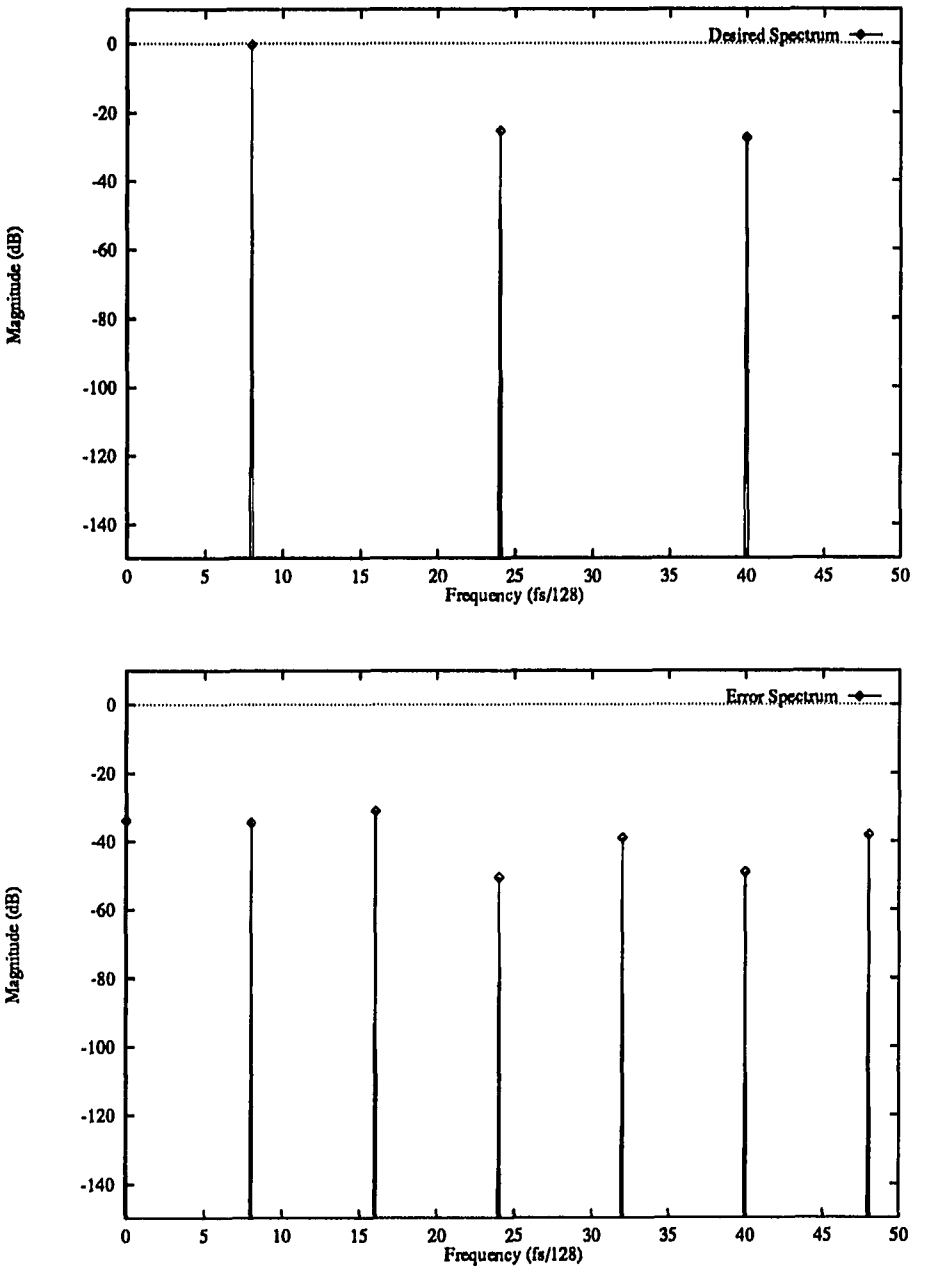

Figure 1.13: Spectrum of the desired (top) and error (bottom) signals from simulation (note the quantization errors in the desired signal)

Table 1.1: Table of time-domain error amplitudes — "glitch" heights — for a 3-bit DAC (in fractions of full scale) versus transition codes

Destination Code	Code of Origin							
	000	001	010	011	100	101	110	111
111	-7/8	-6/8	-5/8	-4/8	-3/8	-2/8	-1/8	0/8
110	-6/8	-6/8	-4/8	-4/8	-2/8	-2/8	0/8	0/8
101	-5/8	-4/8	-5/8	-4/8	-1/8	0/8	-1/8	0/8
100	-4/8	-4/8	-4/8	-4/8	0/8	0/8	0/8	0/8
011	-3/8	-2/8	-1/8	0/8	-3/8	-2/8	-1/8	0/8
010	-2/8	-2/8	0/8	0/8	-2/8	-2/8	0/8	0/8
001	-1/8	0/8	-1/8	0/8	-1/8	0/8	-1/8	0/8
000	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8

2^{2N} possible transitions (or 65,536 in this 8-bit example).

To get a better feel for the dynamic range limitations that data skew can impose, a short survey of signals and their SFDR's will be made here. In all cases the sampling frequency is 128MHz (but it is only the relationship between f_o, ξ, and f_s that is of concern), ξ is -1/64 of the clock period (on precedes off), and the DAC has 12-bit resolution, unless otherwise noted. The desired signals contain quantization errors, are represent what the signal would look like without data skew errors. The error signal is generated by the spectrum of the error pulses whose dimensions are determined by ζ and the transition codes. Figures 1.14 and 1.15 show the desired and error signals in both the time and frequency domains for a full scale 250KHz sine wave ($f_o = f_s/512$), while Figures 1.16 and 1.17 are at 2MHz ($f_o = f_s/64$).

It is interesting to notice that the error patterns of these signals and those of the 3-bit DAC look similar in the time domain. The reason for this is that the largest glitches occur when the signal crosses what is called a *major carry*, or where one of the large bit currents changes state. There is one half-scale carry (at mid-range), two quarter-carries (1/4 and 3/4 of full scale), four eighth-carries, and so on. The reason the patterns are not identical is that the exact amplitude of the error pulse is determined by the exact transition, but these variations are usually small with respect to the total. This means that glitch is only a very weak function of the

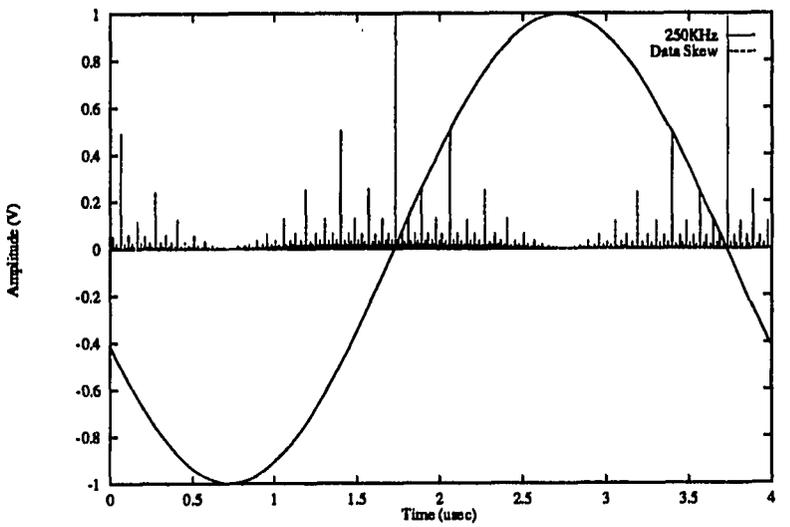

Figure 1.14: Simulation of desired and error signals at 250KHz for a 12-bit DAC (fs/fo = 512)

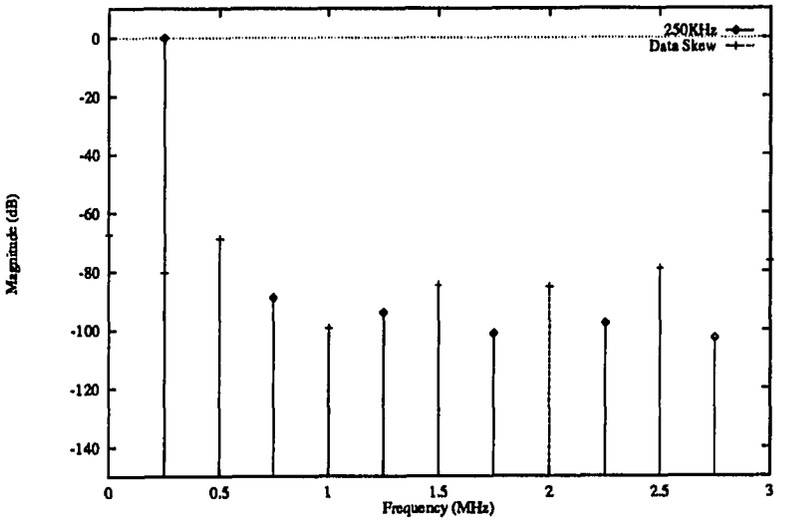

Figure 1.15: Simulation of desired and error spectrums at 250KHz for a 12-bit DAC (fs/fo = 512)

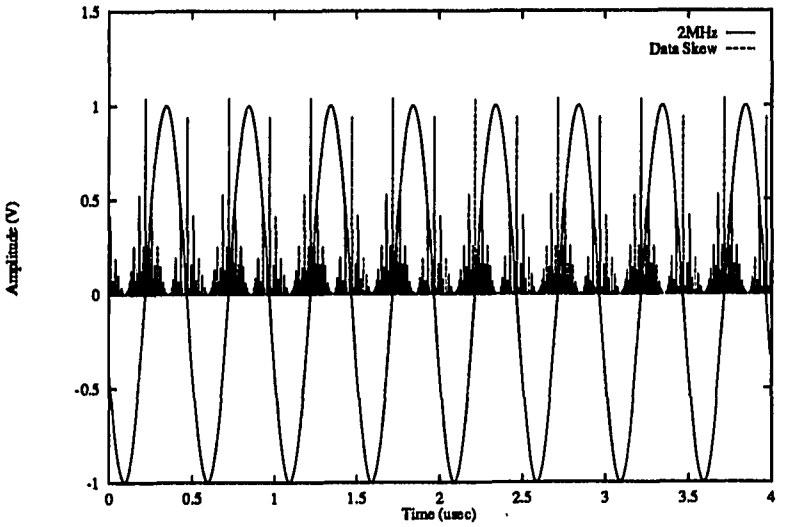

Figure 1.16: Simulation of desired and error signals at 2MHz for a 12-bit DAC (fs/fo = 64)

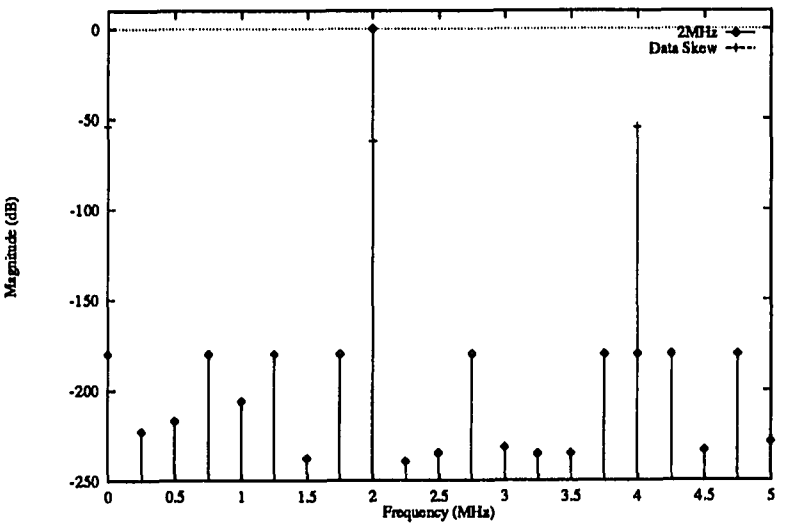

Figure 1.17: Simulation of desired and error spectrums at 2MHz for a 12-bit DAC (fs/fo = 64)

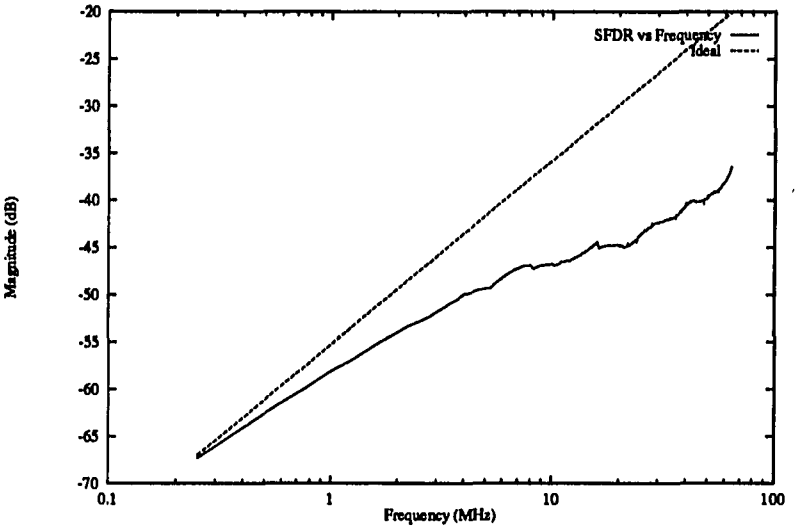

Figure 1.18: Plot of spurious-free dynamic range versus output frequency from simulation (The ideal 6dB/octave line is provided for reference only and is not intended to imply a theoretical value)

number of bits.

Another implication of this pattern is that all full-scale sine, triangle, and square waves will cross these transitions the same number of times per period, but the total number of transitions per period will vary with the frequency of the signal being produced. As a sine wave increases in frequency, the pattern of the harmonics will change little until fairly high frequencies are reached, where multiple carries will be crossed in the same transition. As Figure 1.18 shows, the magnitude of these spectral components will grow because the same errors become a greater portion of the total signal (as the period shrinks, the error density goes up). This plot was made by performing simulations at 256 linearly spaced signal frequencies (which will change the relationship between ζ and f_S), and plotting the results. Obviously the spectral pattern will vary depending on the type of signal being generated (sine, triangle, square, multi-tone, etc.), but the same trends should hold for each.

For the same reasons, the phase of the signal should have little impact on the spectral errors, but both the amplitude and DC offset do because they determine

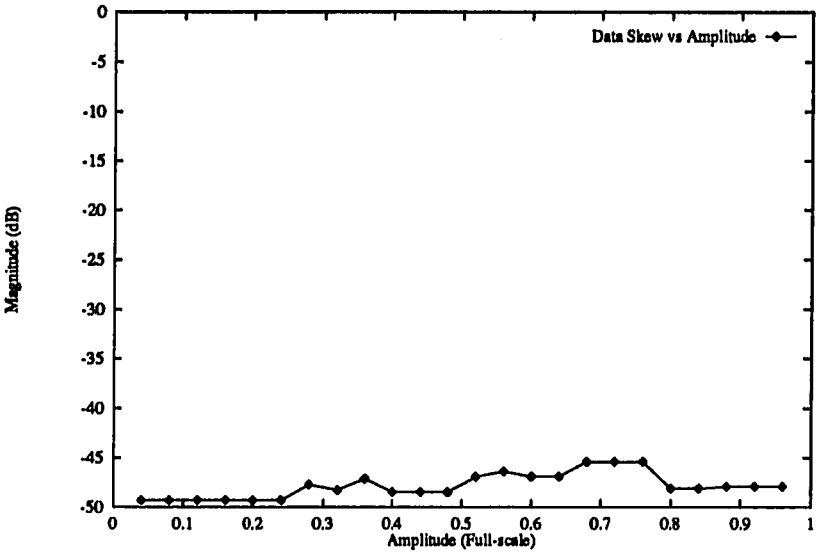

Figure 1.19: Simulation of data-skew error magnitude versus sine wave amplitude

which and how many of the major carries are crossed (Figures 1.19 and 1.20). The points at which the data was taken for these graphs are emphasized on the plot, and the frequency of the signal is 10 MHz in both cases. Note that the SFDR for the errors of Figure 1.19 would fall steeply as the amplitude drops.

Two simulations round out this survey of data-skew errors. Figure 1.21 is an example of a multi-tone signal while Figure 1.22 is a ramp. In the case of arbitrary, multi-tone signals, the spectrum will vary significantly with the relative phases of the tones, because this can strongly effect the number of carry transitions per signal period.

Not all data-skew errors are due to a uniform delay in all of the ongoing transitions. There can be a relative delay between different bit cells due to transmission line effects or varying time constants inside the cells. In general, these errors will be an order of magnitude smaller than the rise and fall errors presented here, because the associated parasitics are much larger than the mismatch of the routing time constants.

There are a number of other sources of glitch in addition to data skew. Nonlinear

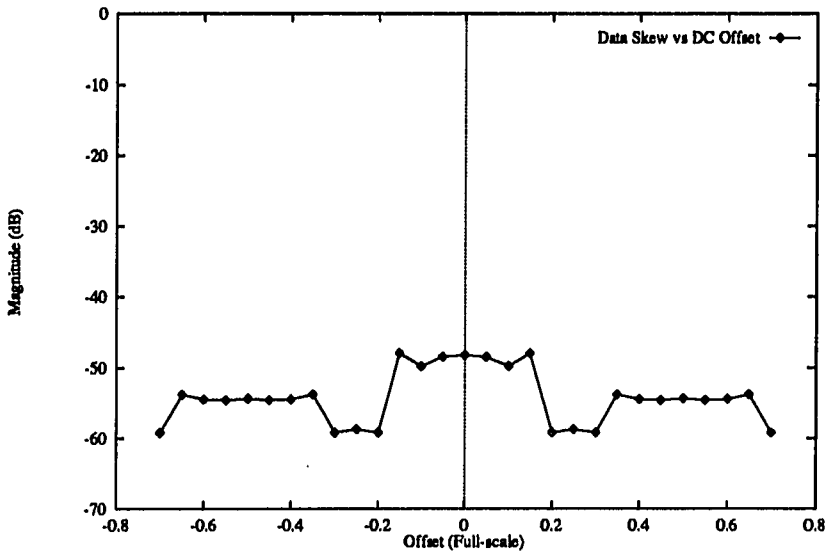

Figure 1.20: Simulation of data-skew error magnitude versus DC offset (signal amplitude is 20% of full-scale)

capacitance and resistance at critical nodes can cause voltage-dependent settling, which is a transition error with spectral impact. Digital feed-through and switch charge also contribute to glitch errors.

Although there are a number of circuit-level cures to these and other forms of transition glitch discussed in Chapter 2, segmentation and thermometer decoding will also be covered here for the sake of continuity. Full segmentation is a DAC construction technique where all of the bits consist of binary multiples of a unit (or least-significant bit - LSB) current cell. This is in contrast to the strict binary-weighted DAC, all of whose cells are of different size (merely a layout distinction). In a fully-segmented, thermometer-decoded DAC, each of the unit currents is controlled by its own switch and decoding logic such that the total number of units changing state is always proportional to the desired output step. Figure 1.23 shows how each DAC performs the one-LSB transition from 011 to 100. Although decoded DACs still suffer from the turn-on delay of the switches, the amplitude of the resulting glitch is directly proportional to the transition, resulting in a frequency-dependent gain error

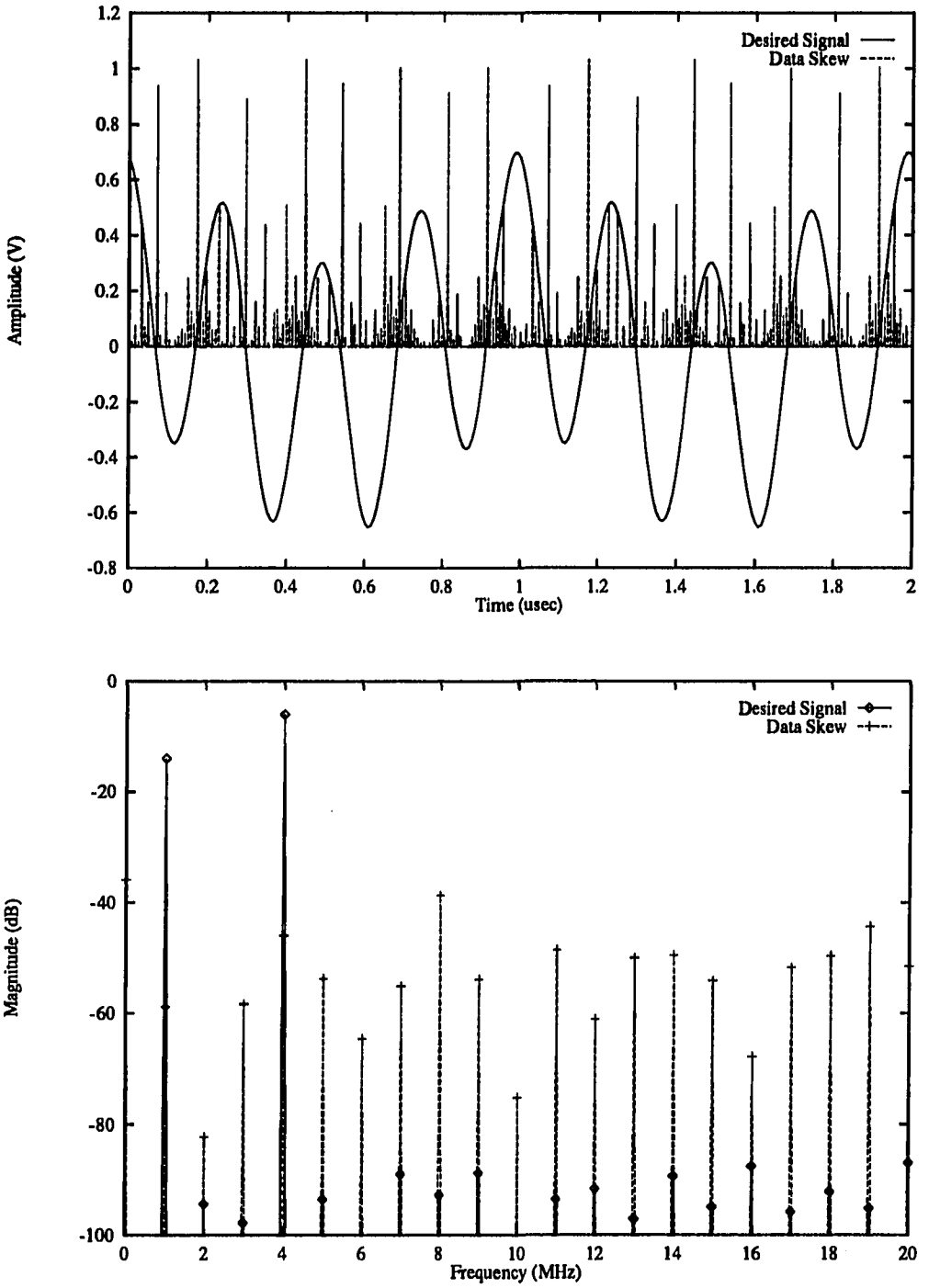

Figure 1.21: Simulated time-domain signal (top) and spectrum (bottom) of a multi-tone signal with data skew

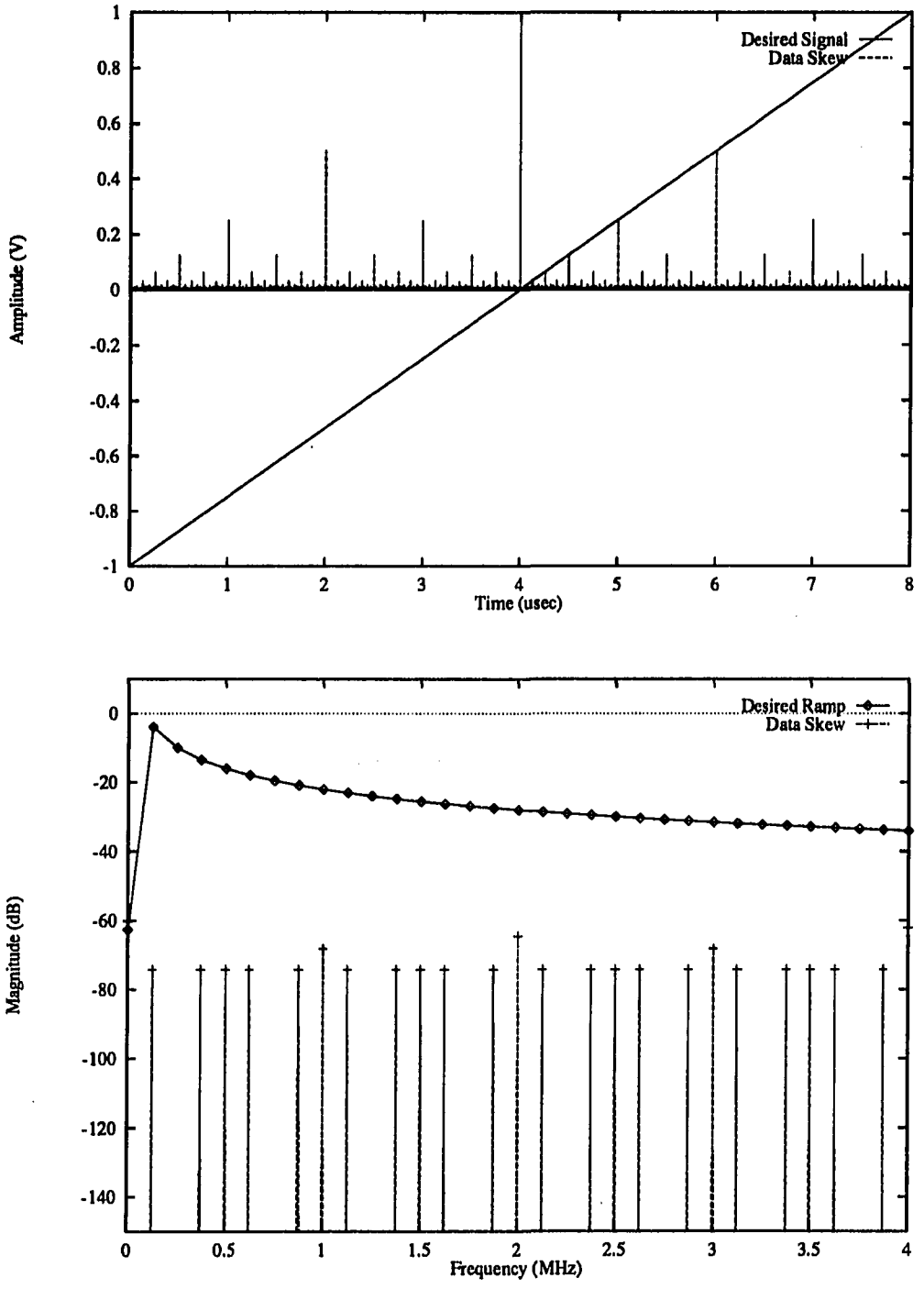

Figure 1.22: Simulated ramp spectrum with data skew

similar to that of a linear system.

Many modern DACs are only partially decoded/segmented, leaving the low-order bits binary-weighted. This has many of the advantages of decoding without enormous hardware penalties (number of switches, decoding logic, etc.). For example, if only the MSB is decoded, the major half-scale carry glitch is cut in half, to the size of the quarter-carries. If the top two bits are decoded, there are 7 eighth-carries, and so on. Thermometer-decoding the high bits thus removes the worst of the glitches.

1.8.5 Jitter

Jitter is usually the result of noise in the clock generator or threshold uncertainty in the clock buffers and other gates. This clock phase noise, in a DDS system, can vary the position and width of the pulses generated by the converter and sampler. Because jitter is a random process, it is most commonly measured as an RMS phase variation about the expected value. Like other random variables, it can be described by a probability density function (PDF) that indicates the likelihood of the clock transition at different times.

The main question here is how does this high-frequency variation of pulse width and position affect the spectral purity of the signal being generated. As with glitch, the distortion output signal can be decomposed into an ideal and error pulse train (Figure 1.24). The amplitude of the error pulses is determined by the amplitude of the ideal signal, but its width, position, and sign are a function of the PDF of the jitter. Note how the sign of the jitter (together with the sign of the transition) determines the polarity of the error pulse and its position relative to the proper clock edge. The absolute value of the error pulse amplitude is fixed by the step size taken by the signal. The width of the pulse is determined by the absolute value of the jitter.

To effectively calculate the spectral impact of jitter, its statistics must be combined with a model similar to that used on data skew, but a derivation will not be included here.

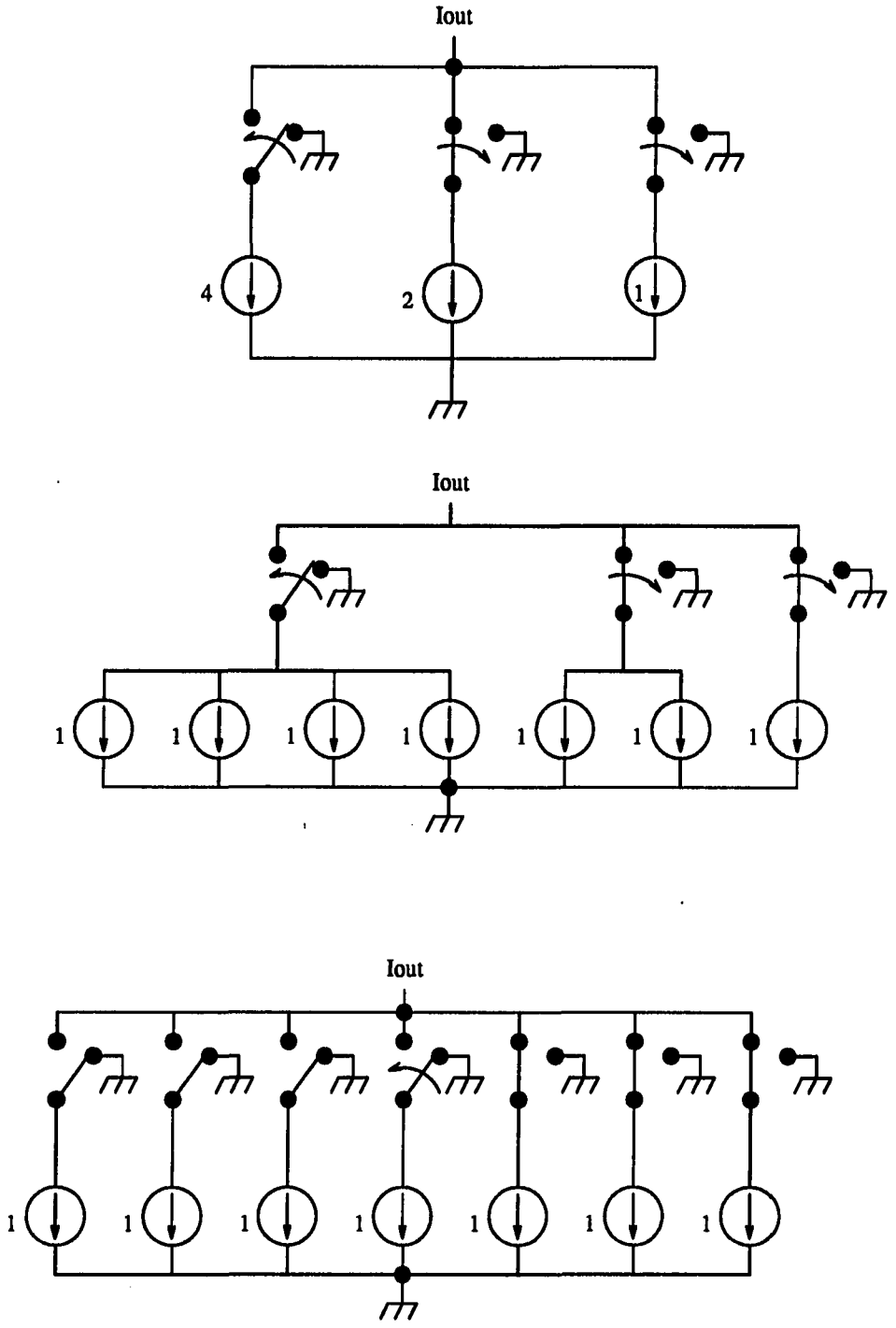

Figure 1.23: Major carry transitions in binary (Top), segmented (Middle), and fully-decoded (Bottom) DACs

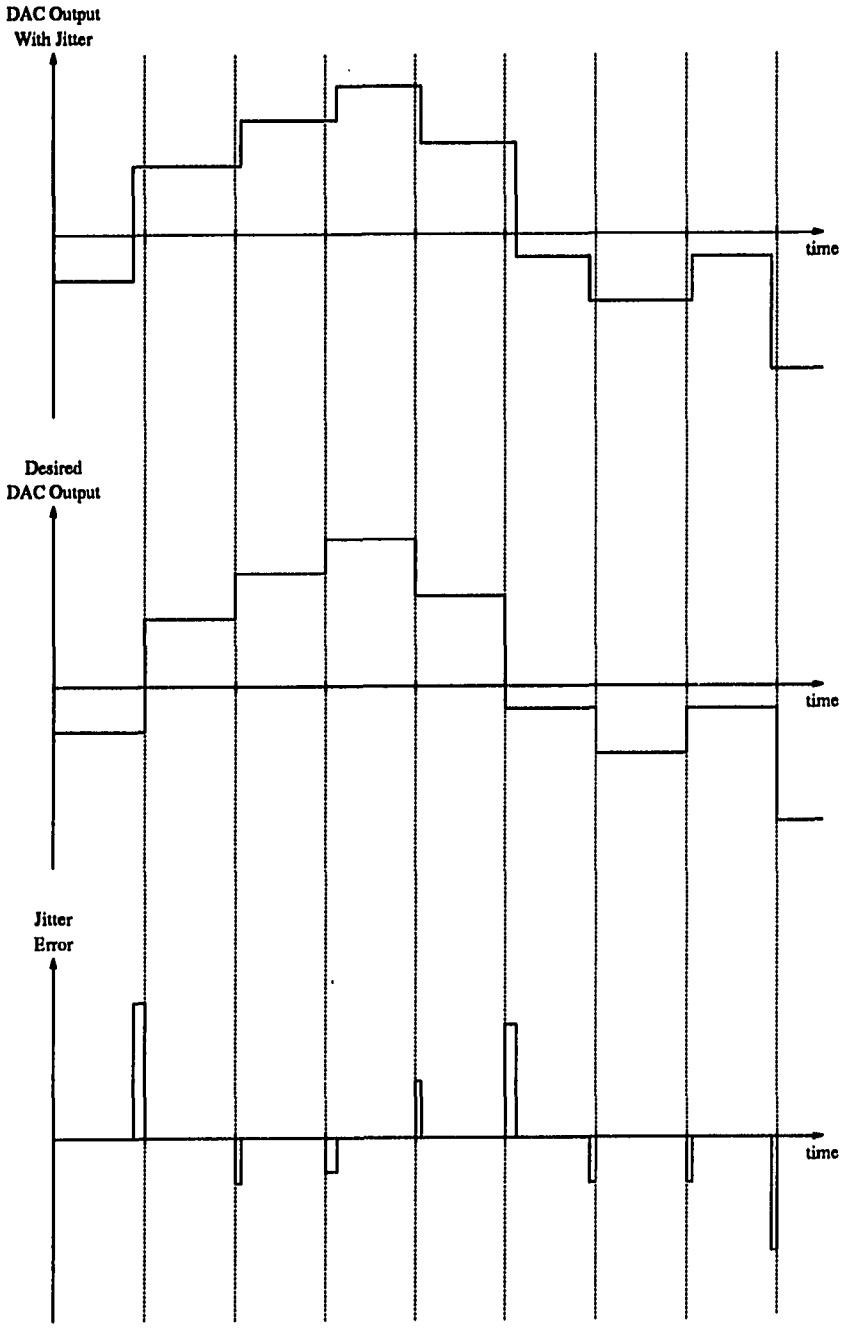

Figure 1.24: Decomposition of jitter-distorted signal into the ideal signal and error pulse train

1.8.6 Noise

There are many classes of noise that must be considered in the design of a signal-generation system employing digital synthesis. In addition to the earlier-mentioned quantization noise (which is a phenomenon unique to signals generated or controlled by digital techniques), there is circuit noise generated by the resistors and active devices of the analog blocks, interference from the clock or digital lines, etc.

In general the cumulative effects of all noise sources will be of concern and will be represented by the RMS value at the output, but for certain applications the noise in a certain frequency band will be of interest. In such cases, the noise figure or spectral density will be specified. In many cases, however, noise is of less concern than additional spectral components.

1.9 Sampler Architectures

Perhaps the most important system-level consideration for DDS signal generation is the architecture of the sampler, or, in fact, whether to sample the DAC output at all. The main advantage of sampling lies in the fact that by blocking the settling of the DAC, the low- and high-frequency design problems can be decoupled. Low-frequency, or "DC" linearity specifications – those that remain even when the clock and signal frequencies are low – are the main concern of the DAC. Measures of high-frequency linearity like SFDR and THD, which are usually governed by transient response and degrade with increasing clock and signal speed, are the design constraints of the sampler. The DAC still has to have fast settling so as not to limit the speed of the system, and so that any nonlinear transients are gone before sampling. It is shown in Chapter 2 that fast-settling and linear transients are conflicting design requirements in a large, highly-coupled structure like a DAC. Likewise, the sampler must not degrade the DC performance of the DAC, but Chapter 2 also shows that this does not conflict with the requirement of fast, linear settling in a simple component like a sampler [7, 24].

There are basically two types of samplers: those that maintain their values after sampling, and those that Return-To-Zero (RTZ). In the first category are Sample-and-Holds (SH) and Track-and-Holds (TH). Both of these architectures rely on a switch

and a capacitor to hold the sampled value. It turns out that the time constant formed by these components helps determine both the speed and dynamic linearity of the structure [25, 26], generating conflicting design goals. A small capacitor and wide (low resistance) switch will sample quickly due to the small time constant they form, but the errors due to charge dump from the switch are proportional to the switch capacitance (also dependent on width) over the hold capacitance.

Frequently an RTZ structure is simply a switch that connects the output of a current-source DAC to the sampler output or to ground, as shown in Figure 1.25. In many cases, RTZ's are called mixers because, when they are operated at very high frequencies, the switches do not have a chance to turn all the way on or off. The sampler then begins to act like a variable attenuator, as if the control signal which modulates the DAC output were no longer a square wave.

There are some drawbacks to using a sampler. The update rate of the system, which controls the maximum frequency signal that can be generated, suffers by almost a factor of two because more operations must be completed in a single clock period. Also, unless a sample-and-hold structure is used, the narrowing of the output pulses will drop the output signal amplitude. This should not affect the dynamic range since most signal components will be affected identically (with the possible exception of noise). By adding an additional component to the system, the complexity of the clocking and related controls will be increased, but this may be balanced by a simpler DAC design. It is also possible that the sampler may remove the need for de-skewing latches before the DAC, unless it is found that the DAC settling time suffers without them.

Another consequence of a return-to-zero DAC is that the transmission zeros are moved to higher frequencies, causing less natural attenuation of the aliased signal components. For a staircase DAC, the sinx/x envelope drops the amplitude of a signal at $f_S/2$ by 3.92dB, but the relative drop (from DC) in a sampler with half the pulse width of the DAC is only 0.91dB. This makes the filtering problem more complex but the sinx/x compensation easier.

The movement of these zeros also opens up another opportunity. If the anti-imaging filter is replaced with a bandpass structure, it is possible to generate signals higher than the Nyquist rate. It must be remembered that separating a single aliased

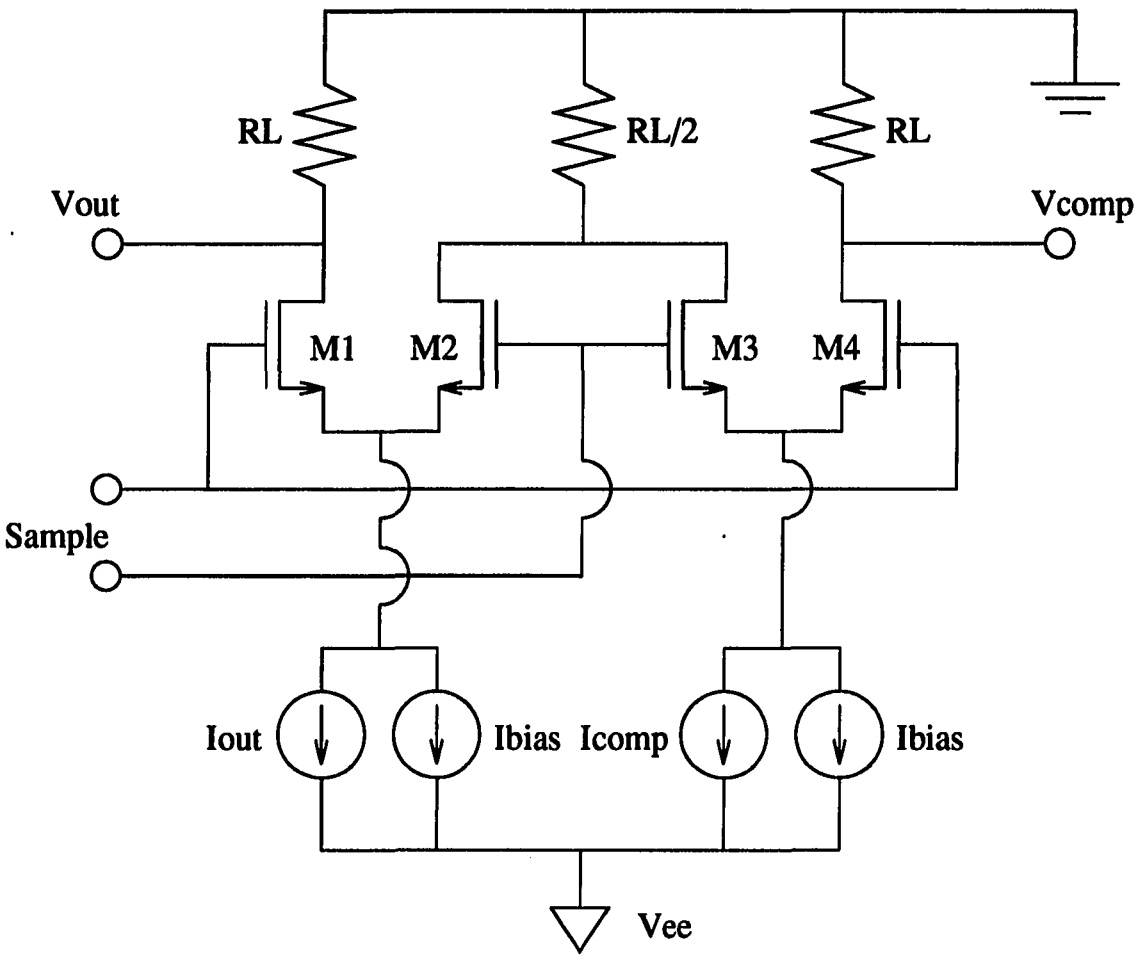

Figure 1.25: A possible return-to-zero sampling structure

component from the others and the baseband signal is an even more significant problem because, in addition to being a higher frequency filter, its effective guard band is reduced. For example, to generate a signal at $\frac{f_S}{2} + \epsilon$, there is also a component at $f_S - (\frac{f_S}{2} + \epsilon) = \frac{f_S}{2} - \epsilon$, only 2ϵ away. The settling poles also give an additional low-pass roll-off that eventually limits the viability of this scheme for very high-frequency generation.

Another advantage of sampling is that it adds a measure of predictability to the error signals that does not exist in DACs due to their highly-coupled nature. These advantages are discussed in Chapter 2.

2. CIRCUIT-LEVEL LIMITATIONS TO DYNAMIC RANGE

The last sections of Chapter 1 established system-level implications of DAC errors and provided mathematical models to map those errors into the frequency domain. This chapter discusses the circuit-level origins of these errors, derives magnitude information for use in the models, and investigates circuit-level cures.

In many of the following theoretical calculations, one-half a least-significant bit (1/2 LSB) is used as a limit for the allowable error. This is, however, only a benchmark or reference point. In an actual design trying to meet such a specification, all of the error sources must be reduced such that their sum does not exceed the limit. Until these calculations are applied to an actual design, it is unclear what the allowable contributions of the various sources are. Thus the need for a reference point.

2.1 High-Speed DAC Architectures

To achieve the highest-speed DAC in a given process, the data path should be as free of negative feedback systems as possible. This is because to stabilize such a feedback network, some of the basic bandwidth of the system must be sacrificed so that the loop gain of the system is gone before the parasitics are encountered. In an open-loop system the parasitics generally determine the bandwidth directly, independent of system stability.

While avoiding feedback, the parasitic time constants in the signal path must also be minimized. This requirement usually translates into maximizing the transconductance for a given parasitic node capacitance. The structure that naturally lends itself to this is the current-switching DAC, because it consists almost exclusively of low-impedance nodes, while most voltage-based DACs do not.

Current-steering or current-switching DACs achieve conversion by switching var-

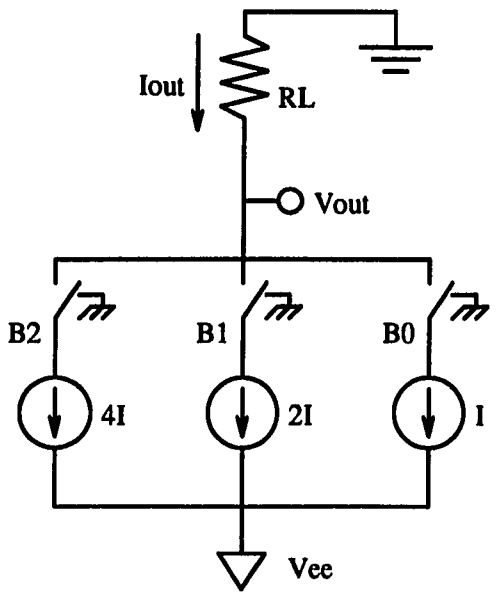

Figure 2.1: Binary-weighted current steering DAC

ious sized currents to or from the output. The simplest case is shown in Figure 2.1 where DAC bits control the direction of binary-weighted currents.

In applications where a voltage output is needed, the current can be fed through a resistor tied to a DC voltage (usually ground) [27, 28]. This differs from a system where an output buffer or amplifier is needed because these components limit the bandwidth of the DAC [29].

2.2 Current-Switching Cell

The most basic component of a current-steering DAC is the switching or "bit" cell. For the purpose of this discussion it includes the current source (whose size is determined by the DAC architecture) and the switching transistors that direct the current based on an incoming digital signal.

Since this discussion takes place at the transistor level, a process must be chosen. For completeness, both bipolar and CMOS techniques are covered, and, where appropriate, the advantages to a merged Bipolar-CMOS (or BiCMOS) process are

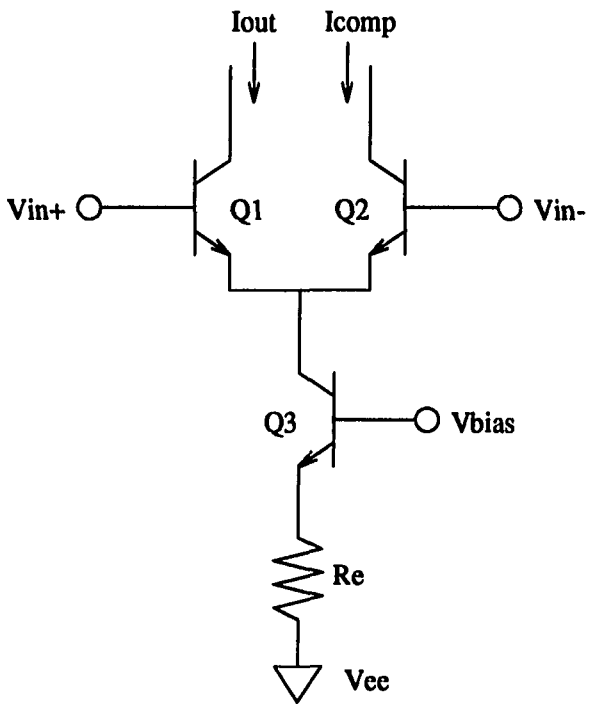

Figure 2.2: Bipolar bit cell

illustrated.

2.2.1 Bipolar

A simple bipolar bit cell is shown in Figure 2.2. The bias voltage (V_{bias}) and the emitter resistor (R_e) set the size of the current, and the differential pair (Q_1 & Q_2), when driven hard, control whether the current is steered to the output or its complement [28, 30, 31].

The differential signal, $V_{in+} - V_{in-}$, necessary to switch the current can be determined by first defining an acceptable current error ratio, ϵ. For the case where $V_{in+} > V_{in-}$:

$$\epsilon = \frac{I_{comp}}{I_{out} + I_{comp}} \approx \frac{I_{comp}}{I_{out}} \tag{2.1}$$

$$= \frac{I_s e^{V_{be2}/V_t}}{I_s e^{V_{be1}/V_t}} = e^{(V_{be2}-V_{be1})/V_t} \tag{2.2}$$

and

$$\Delta V = V_{in+} - V_{in-} = V_{be1} - V_{be2} = -V_t \ln(\epsilon) \tag{2.3}$$

Where V_t is the thermal voltage, and I_s is the reverse saturation current of the base-emitter junction. To place this ratio at $\frac{1}{2}$ an LSB of an N-bit converter (to ensure that fluctuations in ΔV from bit to bit are insignificant):

$$\epsilon = 2^{-(N+1)} \tag{2.4}$$

$$\Delta V = -V_t \ln\left[2^{-(N+1)}\right] \approx 0.69(N+1)V_t \tag{2.5}$$

At room temperature V_t is approximately 26mV, so, even in the extreme case of a 20-bit converter, the differential voltage need only be about 375mV.

The actual control signals V_{in+} and V_{in-} can be handled in a number of ways. One of the simplest is to set one to a fixed reference point and drive the other around that level by $\pm\Delta V$, derived from the logic input [32]. There are two disadvantages to this technique. The first is that the collector of Q_3 sees a different voltage depending on the digital input. Not only must this voltage settle, but it also modulates the collector current by an amount determined by the output impedance set by Q_3 and R_E. The second problem is that this scheme sacrifices an extra ΔV of headroom that could be used elsewhere. A solution is to drive the bases of each Q_1 and Q_2 by $\pm\Delta V/2$ (a circuit designed to do this is presented later in this chapter).

The limit to the bandwidth of a DAC is set by the time needed for the output current of the bit cell to settle. One method to calculate this time uses the transition frequency (f_t) of the switching transistor in an approximation for the time constant of the cell:

$$\tau_{cell} = \frac{1}{2\pi f_t} \tag{2.6}$$

where f_t is the bandwidth of the current gain of the transistor. It is determined by process and bias as [33, p. 129]:

$$f_t = \frac{1}{2\pi\left[\tau_F + \frac{kT}{q}\frac{(C_{je}+C_{jc})}{I_c}\right]} \tag{2.7}$$

If the settling behavior is linear and of first order it will have the form:

$$I_{out}(t) = I_{out}(\infty)\left[1 - e^{-t/\tau_{cell}}\right] \tag{2.8}$$

and the time needed for $I_{out}(t)$ to approach $I_{out}(\infty)$ within N-bit accuracy can be found by equating 1/2 an LSB to the settling error:

$$2^{-(N+1)} = \frac{I_{out}(\infty) - I_{out}(t)}{I_{out}(\infty)} \tag{2.9}$$

Solving for t yields:

$$t_s = -\tau_{cell}\ln\left[2^{-(N+1)}\right] = \frac{(N+1)}{2\pi f_t}\ln(2) \tag{2.10}$$

The problem with this approximation is that it is a linearization about the bias point of the conducting transistor, and does not adequately describe both devices. The switching time is also a function of the time necessary to turn "on" the "off" transistor. This is determined by the base current charging the base-emitter depletion capacitance and then generating the base charge necessary to support minority carrier emitter injection. Once the depletion capacitance is charged, the switching transistors exchange base charge. Only at this point is the base charge of one transistor swapped for that of the other. The transition is completed when the off-going transistor's depletion capacitance is discharged [34, p. 206].

Also ignored here are the effect of capacitance at the emitter-coupled node and the finite impedance driving the bases of the switching devices. For an accurate settling-time calculation, a charge control model is necessary. Due to the complexity of this modeling, SPICE simulations are required. Figure 2.3 shows the output current for the bipolar cell shown in Figure 2.2 when driven with the switching voltages shown in Figure 2.4 (key model parameters for the state of the art process used are

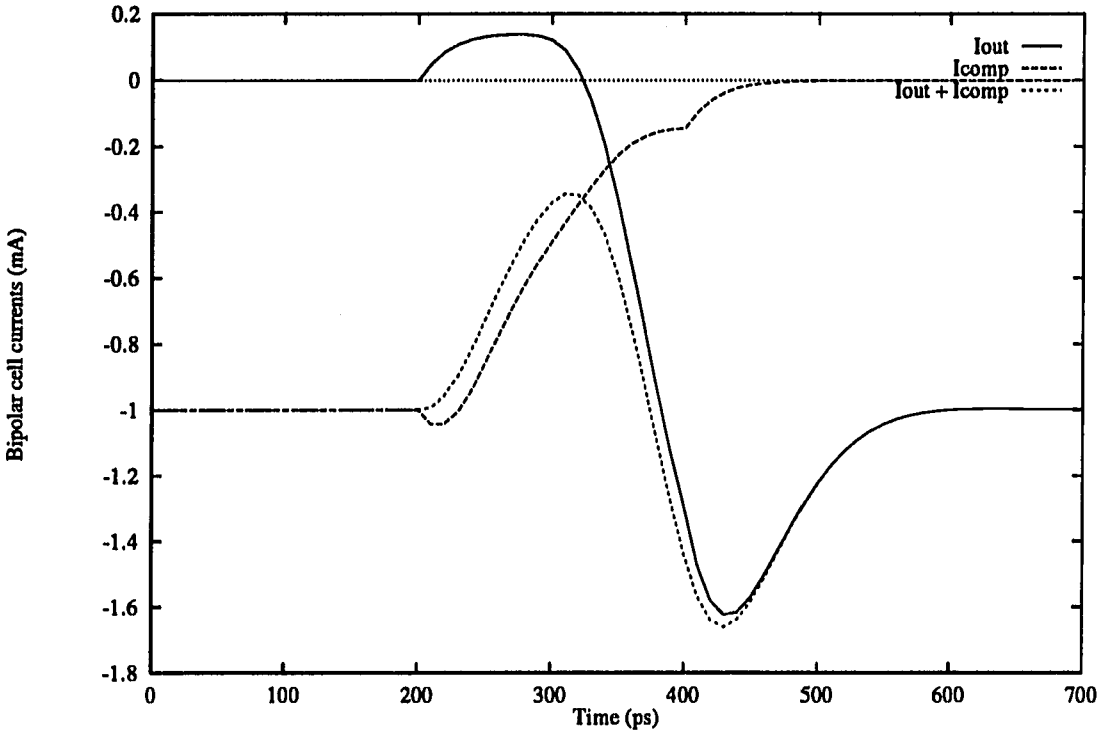

Figure 2.3: Simulated output-current waveforms for a bipolar bit cell

shown in Table 2.1). Notice that the outputs are strongly asymmetrical, leading to data-skew errors. Current settling, even with ideally fast base drives, is to 1, 0.1 and 0.01% in 230, 350 and 420 picoseconds, respectively. Note that this is significantly faster than that achieved with the drives shown in Figure 2.4.

Of particular interest in these simulations is the dip or "bounce" in Q_2's emitter voltage of 120mV (Figure 2.4). It is this transient, and that seen at the base of Q_3 – whose settling is determined by a separate bias loop – that dominates the settling time in virtually all modern, high-speed DACs. Even with zero-impedance biasing the base of Q_3, the voltage across the degeneration resistor (Re in Figure 2.2) is disturbed by about 5mV. These phenomena will be important in determining the magnitude of the DACs dynamic errors.

One of the advantages of the bipolar cell is the low voltage swing necessary to

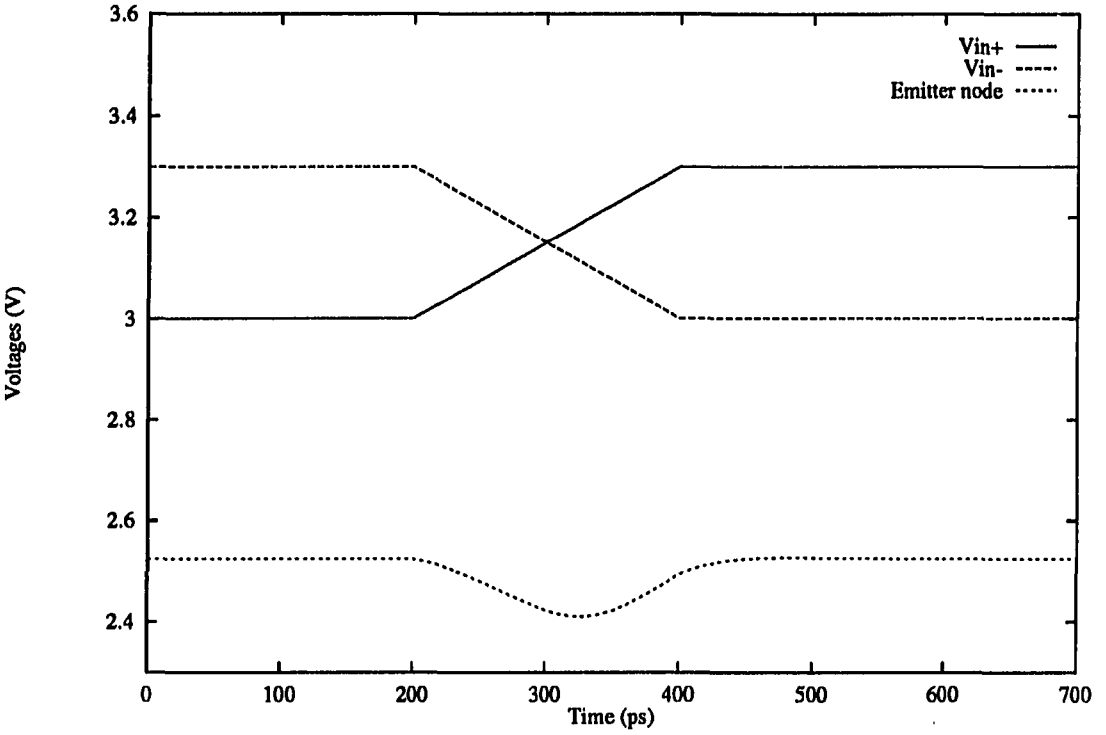

Figure 2.4: Simulated voltage waveforms for bipolar bit cell

Table 2.1: SPICE bipolar model parameters

Parameter	Value	Unit
f_t	6	GHz
BF	60	
RC	100	Ohms
RE	900	mOhms
RB	140	Ohms
IRB	4	uA
RBM	50	Ohms

Parameter	Value	Unit
TF	25	ps
CJS	180	fF
CJC	90	fF
CJE	55	fF
VAF	60	V
IS	42	aA
ISE	5	fA

switch the current. Lower swings limit the charge injection and digital feed-through seen at the output and minimize the disturbance on the bias rails of the cell. This last point will become very important in calculating the system's settling time.

If the digital inputs to the DAC are ECL compatible, then it is possible to drive the bases of Q_1 and Q_2 directly, but frequently the inputs are TTL. A circuit which generates the desired low level swings [35, 36] is shown in Figure 2.5. The core of this circuit is the same as that of Figure 2.2, but now the devices Q_4 and Q_5 shield the switching devices from the full logic levels. The new differential pair steers a current I_x into one of the R_x resistors to generate the ΔV.

2.2.2 CMOS

A bit cell compatible with a standard CMOS process is shown in Figure 2.6. The potential at V_{bias} and the sizing of M_3 determine the current source magnitude, and M_1 and M_2 direct the current to the output or its complement. All devices typically operate in the saturation region to minimize the effects of drain-source voltage fluctuations [27, 37, 38].

Here the differential voltage necessary to switch the current is determined by the sizing of the source-coupled pair and the magnitude of the current. Since the MOSFET's drain current is proportional to the square of the gate-source bias (as opposed to the exponential relationship in the bipolar case), the "off" device in the pair is typically driven into cutoff to ensure sufficient switching. Therefore:

$$\Delta V = |V_{in+} - V_{in-}| = V_{gs} - V_t = \sqrt{\frac{2LI_{bit}}{\mu_o C_{ox} W}} \tag{2.11}$$

where W and L are the width and length of the switching transistors, μ_o is the surface mobility of the channel, and C_{ox} is the capacitance per unit area of the gate oxide. Since this number is typically a half a volt or more, there is less to be gained by scaling down the logic swings before applying them to the gates of the source-coupled pair.

The settling time of a MOS DAC is also best calculated from a charge control model. If the gates are driven with sufficiently fast rise times, the settling is determined by the time needed to evacuate the inversion charge and cut off the channel.

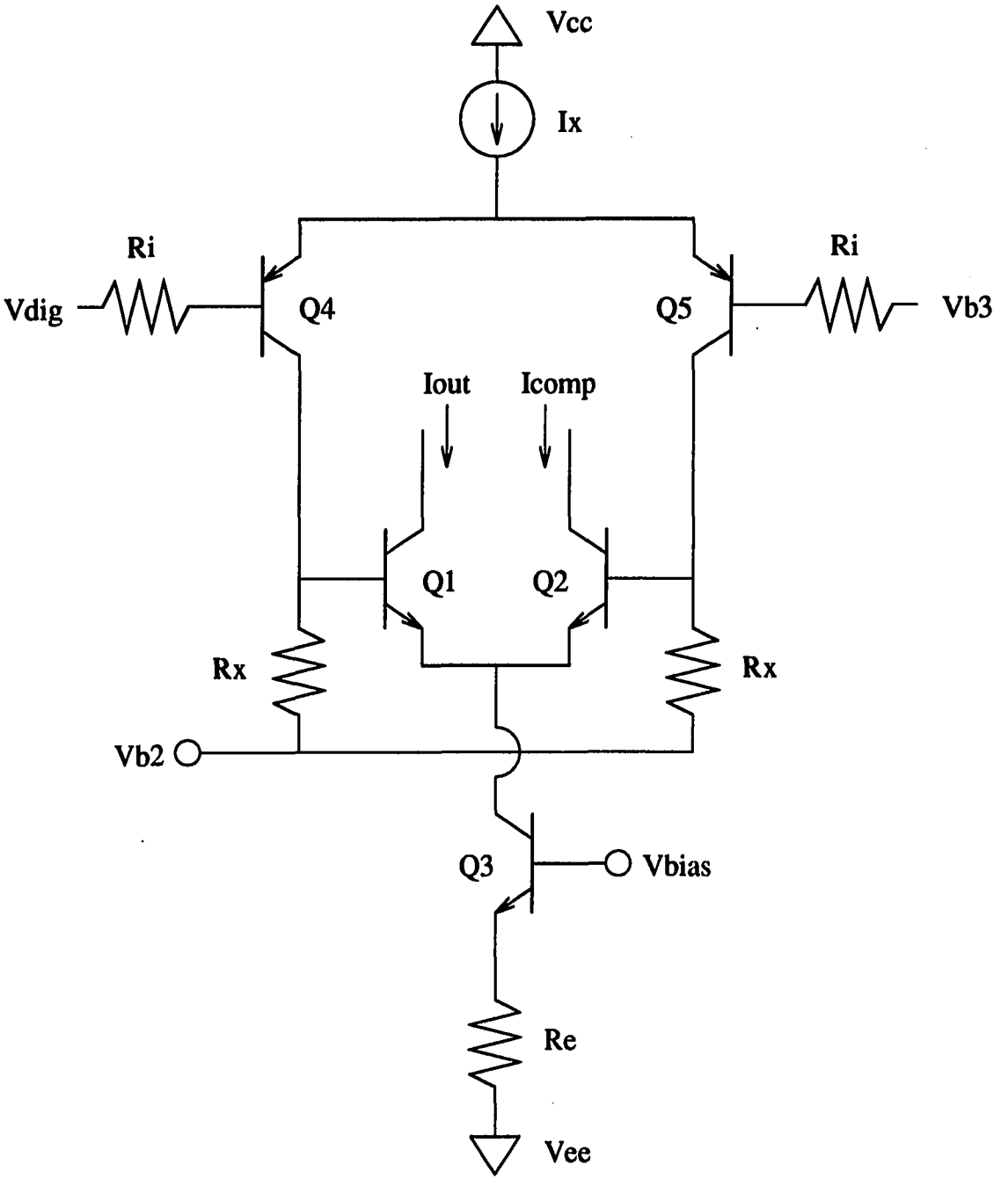

Figure 2.5: A bipolar Craven cell

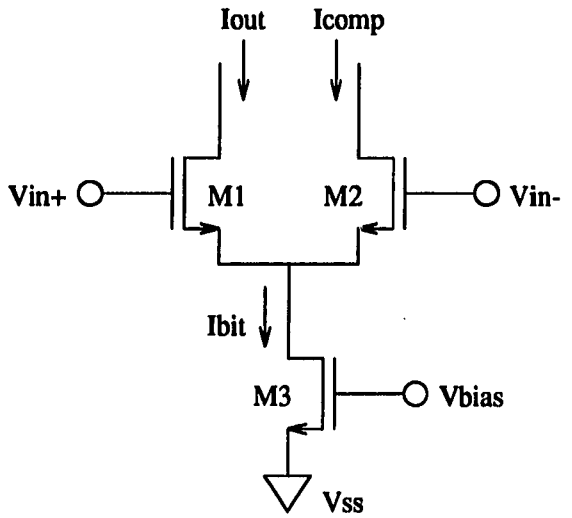

Figure 2.6: CMOS bit cell

Figure 2.7 shows the current switching waveforms of the CMOS bit cell of Figure 2.6 in response to the one nanosecond gate drives of Figure 2.8. Note that the lack of symmetry in the rising and falling current edges, and the disturbance of the source node is similar to what was seen in the bipolar case. When driven by an infinitely fast edge, settling to 1, 0.1, and 0.01% is in 1.9, 2.5, and 3.2 nanoseconds, respectively. As would be expected, these times are almost an order of magnitude slower than the bipolar case. The key SPICE and circuit parameters are listed in Table 2.2.

In contrast to the bipolar case, the MOS devices require a current-dependent drain-source bias to remain in saturation. Due to the lower transconductance-area trade off in MOS technology, this voltage drop tends to be about a volt, drastically limiting the number of devices that can be put in series between the supplies. The voltage "space" that the devices fit into is referred to as *headroom*. Headroom becomes a serious concern when contemplating and sizing cascode devices.

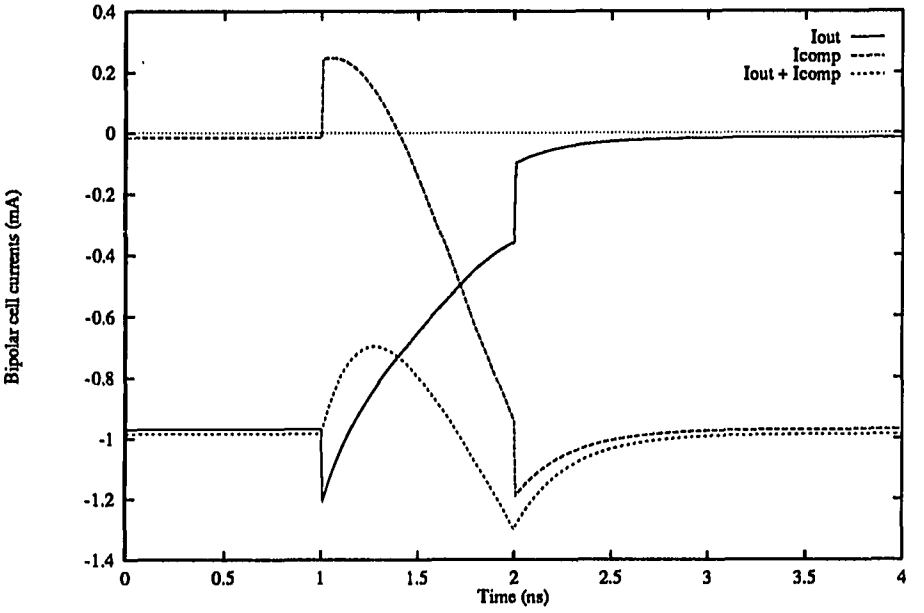

Figure 2.7: Simulated current waveforms for a CMOS bit cell

Table 2.2: SPICE and circuit parameters for CMOS bit cell simulation

Parameter	Value	Unit
I_{bit}	1	mA
ΔV_{gs}	800	mV
W	250	uM
L	5	uM
CJ	300	uF/M^2
TOX	50	nM

Parameter	Value	Unit
lambda	20	m
VTO	800	mV
LD	600	nM
CGD	400	pF/M
CGS	400	pF/M
CGB	100	pF/M

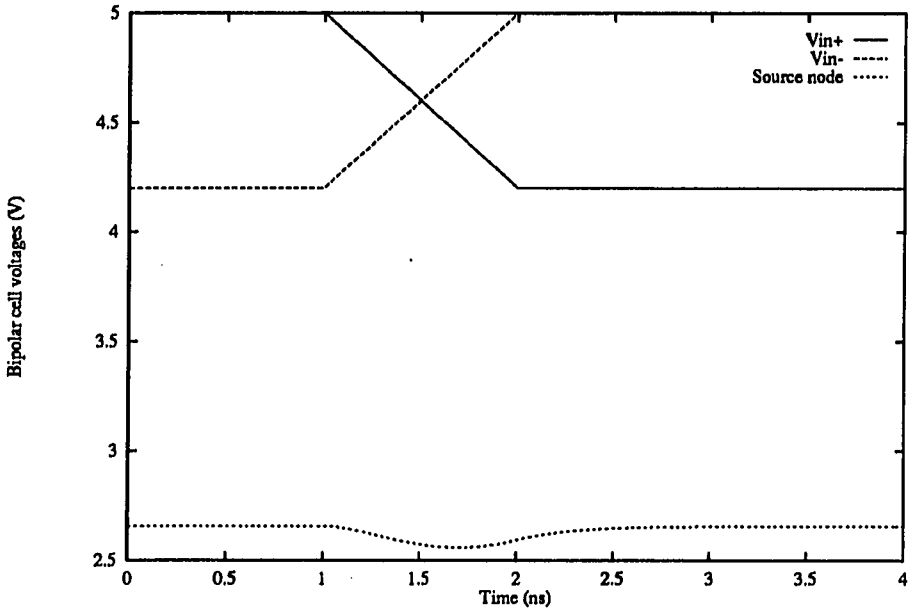

Figure 2.8: Simulated voltage waveforms for a CMOS bit cell

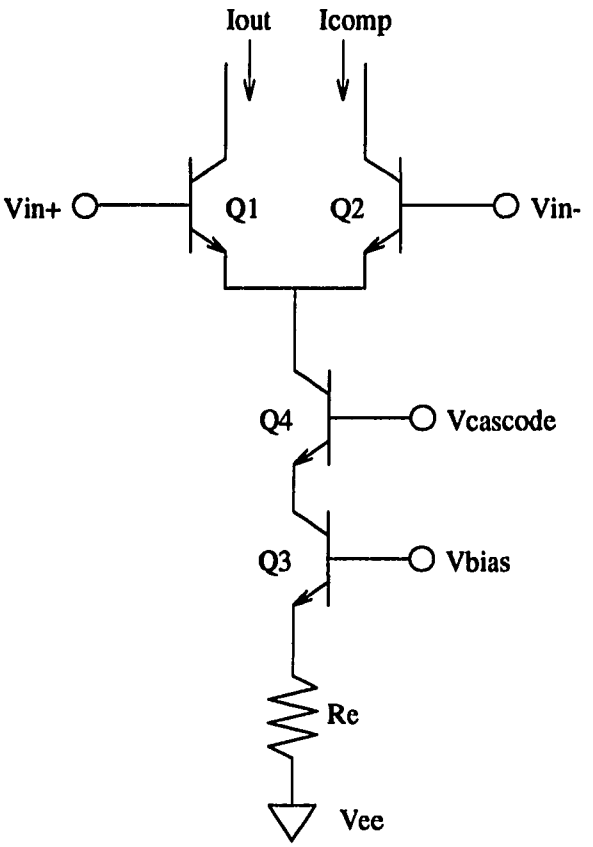

Figure 2.9: A bipolar bit cell with a cascode transistor for disturbance rejection

2.2.3 Cascode current source

Later in this chapter it is shown that the settling time of a DAC is a strong function of the disturbance rejection and bandwidth of the bias voltages. It is especially important to shield the bias rail that sets the current amplitude. To do this it is often necessary to cascode the current source transistor, before connecting it to the switching devices, by inserting a common-base (or common-gate) transistor, as shown in Figure 2.9. In addition to the disturbance-rejection abilities of this cascode, it also increases the output impedance of the source, minimizing the current fluctuations due to switch bias variations and emitter bounce [27, 30].

Also, a cascode inserted between the bit cells and the output will isolate the load from the collector-base capacitances of all of the switching transistors [31]. This

reduces the nonlinear load capacitance, which sets the output time constant and contributes to the distortion (see output sampling).

There are, however, a number of drawbacks that must be considered. The first is the reduction in circuit headroom, especially in a CMOS design. The second is the additional transistors, with the associated area, biasing, and layout complexity.

2.3 Cell Biasing

2.3.1 Reference amplifier

The most important concern in DAC settling time - far more significant than the current cell - is the design of the biasing circuitry which maintains the correct current amplitude. The only successfully established way to achieve the necessary accuracy over temperature and process is to use a feedback structure like that shown in Figure 2.10. This loop works because the amplifier drives V_{bias} in such a way that the $I_{REF}R_{REF}$ product equals a stable reference voltage, V_{REF} [39]. This V_{bias} is the same as that used to establish the current in the basic cells (Figures 2.2 and 2.9). The bit cells then rely on device matching to maintain accuracy between I_{REF} and the bit currents. The bias transistor and R_e of the reference loop must match Q3 and R_e in each of the bit cells. This technique assumes temperature stability for R_{REF} and V_{REF}, high input-impedance for the amplifier, and high loop-gain. Errors in these assumptions will result in DAC gain errors.

This simplified diagram ignores a few key factors. One is that the collector voltage of the reference transistor of Figure 2.10 will not match those of the bit cells, nor does this loop compensate for the base current that flows into the switching transistors. These problems can be corrected by inserting a dummy transistor to resemble the "on" transistor in the switching pair. Note that this transistor's bias need not be set by the reference loop because its variation has only a small second-order effect on the output current.

2.3.2 Disturbance rejection

The most significant contributor to DAC settling is the time needed for the reference loop to resettle after a disturbance. This time is related to both the magnitude

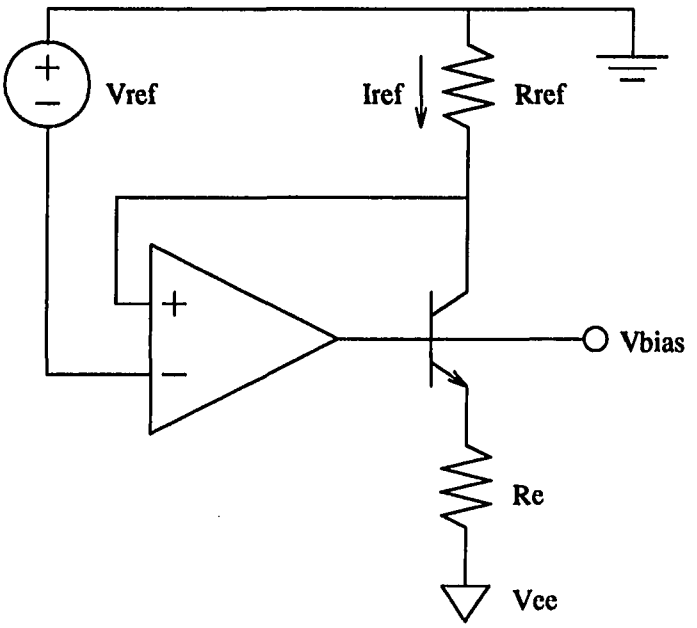

Figure 2.10: Reference amplifier loop for establishing correct current magnitude

of the disturbance injected into the loop, and the closed-loop bandwidth.

The magnitude of the disturbance is determined by the coupling mechanism and the impedance seen at the point of injection. If we assume that the main disturbance is due to the collector-base capacitance of Q_3 in Figure 2.2, then its magnitude is determined by the amount of the emitter bounce as:

$$d = \frac{\Delta V_e Z_L}{Z_L + \frac{1}{sC_{cb}}} \tag{2.12}$$

where d is the disturbance magnitude, Z_L is the impedance of the loop, ΔV_e is the emitter bounce, and C_{cb} is the collector-base capacitance. Assuming a single pole response for the amplifier and a unity-gain feedback network ($R_E = R_{REF}$), the loop simplifies to the small signal circuit of Figure 2.11, where R_L and C_L are the open-loop impedances at the amplifier output, and g_m is the amplifier transconductance. Note that for this analysis, ΔV_e is the input. The closed-loop impedance is then:

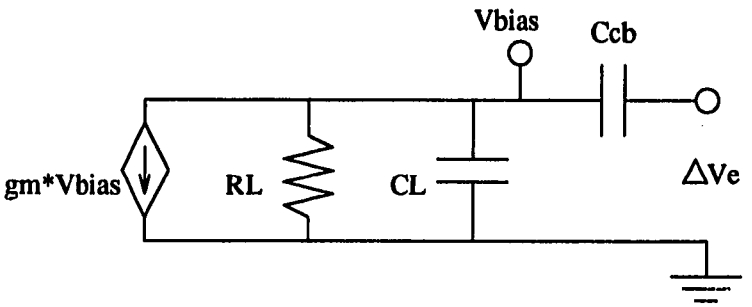

Figure 2.11: Small-signal equivalent circuit for the reference loop during disturbance injection (ΔV_e is the input)

$$Z_L = \frac{R_L}{1 + g_m R_L + sC_L R_L} \tag{2.13}$$

which at high frequencies (if ΔV_e is a fast edge) looks like:

$$Z_L \approx \frac{1}{sC_L} \tag{2.14}$$

Therefore:

$$d \approx \Delta V_e \frac{C_{cb}}{C_{cb} + C_L} \tag{2.15}$$

The closed-loop time constant is:

$$\tau_L = C_L / g_m \tag{2.16}$$

yielding a settling time to 1/2 LSB of:

$$t_s = \tau_L \ln\left[d2^{(N+1)}\right] \tag{2.17}$$

$$t_s = \frac{C_L}{g_m} \ln\left[\Delta V_e \frac{C_{cb}}{C_{cb} + C_L} 2^{(N+1)}\right] \tag{2.18}$$

It can also be shown that the DC accuracy of the loop is proportional to the loop gain, or $g_m R_L$. This fact, and Equation 2.18 indicates that in order to maintain both high DC accuracy and fast settling, the loop transconductance should be high.

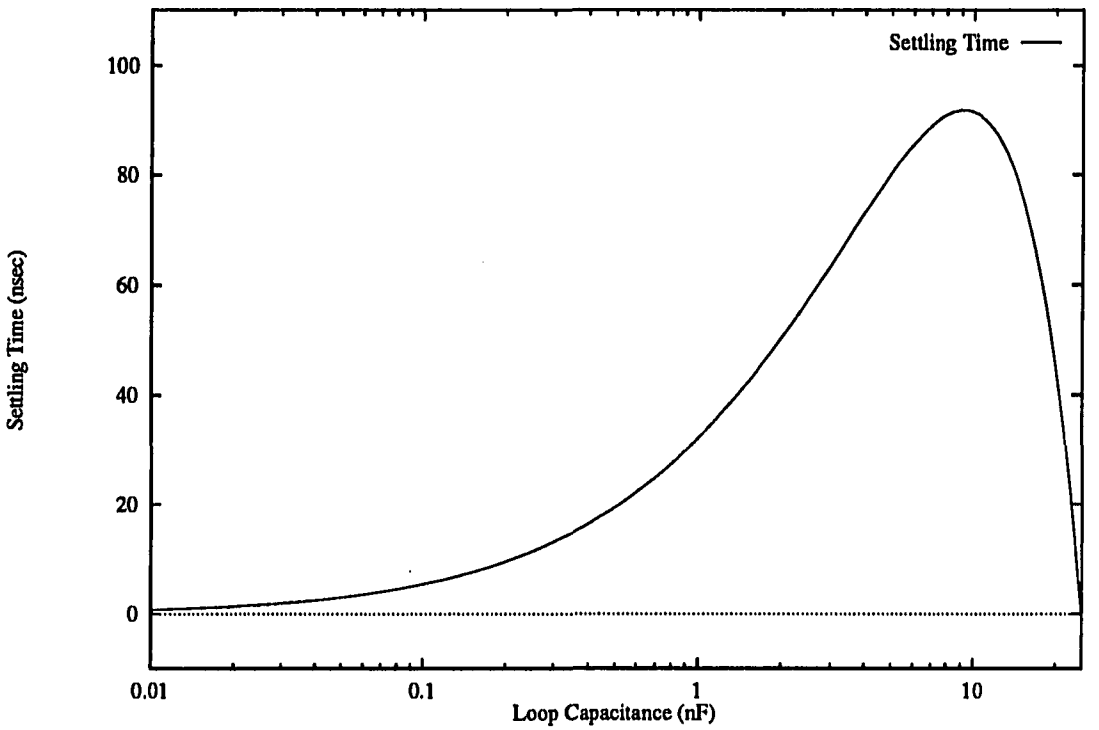

Figure 2.12: Plot of disturbance settling time versus loop decoupling capacitance

Notice that applying a large capacitor at the base rail (in an effort to minimize the disturbance) is effective only if the quantity inside the natural log function can be made to approach one. If this happens, the loop settling time goes to zero because the magnitude of the disturbance is less than one half of one least-significant-bit. At this point the bandwidth of the bit cell limits settling. Figure 2.12 is a plot of settling time versus loop capacitance for $N = 12$, $\Delta V_e = 0.3\text{V}$, $g_m = 0.1\text{A/V}$, and $C_{cb} = 10\text{pF}$.

Unfortunately, to achieve "zero" settling time in the loop, the C_A/C_{cb} ratio must be quite large, especially for high-resolution converters. This is virtually impossible to achieve on-chip because C_{cb} is really the effective collector-base capacitance of all the bit cells. Off-chip capacitors bring a host of complications, including bond-wire inductance and compromised loop stability.

Going off-chip also makes the loop more susceptible to external disturbances, and if they are larger than 1/2-LSB they will cause a nonzero settling time. With the bandwidth compromised by the decoupling capacitance, the new settling time may be quite long (notice the steepness of the right-half portion of the curve in Figure 2.12). For high resolution converters it is therefore best to maximize the loop bandwidth by minimizing C_L and maximizing g_m. Unfortunately this frequently makes loop stability and high loop-gain harder to achieve.

Before leaving disturbance-rejection issues, it should be emphasized that only the current-source bias point should be set by the reference loop. The reason for this is that for every extra connection to the loop, there is another disturbance mechanism, and another feedback path, both of which could slow settling. Fortunately there is little loss in output accuracy when setting cascode and switch level biases from nonfeedback circuitry.

2.4 Static Linearity

Although static (or DC) linearity is absolutely essential to distortion-free signal generation, techniques for achieving it are well understood and no longer limit the spectral purity of high-speed DACs. The origins of static nonlinearities are also varied and plentiful. For these reasons, and because of the wealth of published material on the subject, only a brief discussion, concentrating on a few design techniques, is presented here.

With few exceptions [40], uncalibrated and untrimmed current-source DACs rely on the active and/or passive device-matching of the fabrication process to achieve the necessary linearity [41, 42, 43]. Layout [44], interconnect resistance [27], mechanical stress, thermal and process gradients [44], as well as area and cost trade-offs, must all be carefully considered in such a design.

Two design techniques which reduce the impact of many error sources simultaneously are segmentation and thermometer decoding. Chapter 1 discussed their benefits over data-skew and transition errors, but they also improve both integral and differential linearity. Interdigitation and common-centroid layout minimize the effects of various process and thermal gradients, and are very compatible with segmented

designs.

Trimming, a method of adjusting the transfer function by permanently altering the device characteristics after fabrication, can produce results unachievable through raw device matching. The size and complexity of such a DAC design can also be far less, but aging, drift, and post-packaging stresses must still be considered, along with the cost of trimming [45, 46, 47, 48].

Self-calibration, or the ability for a circuit to measure and correct for its own errors [49, 50, 51], has allowed fairly crude, inexpensive processes to compete in high-end markets. Such designs have periodic "calibration cycles" when the errors are corrected. The time between calibration varies widely with the design, but in some cases occurs only at power-up. Although calibrated systems are frequently more complex than trimmed devices, they do not suffer from the same long term effects (aging, etc.).

2.5 DAC Transition Errors

As the update rate of a converter is increased, the relative amount of time the output spends in transition (settling) is increased. It was mentioned in Chapter 1 that if this transition is nonlinear it will contribute to distortion in the output signal. Since these errors do not show up in DC DAC tests, they are frequently referred to as dynamic nonlinearities. It is these errors that limit the spectral performance of high-speed DACs.

2.5.1 Data skew

Ideally, when a DAC transition occurs, all of the switches are thrown at the same time, and the output changes state instantly. In real DACs, however, there are always delays between individual cell transitions. For example, at what point in Figure 2.3 does the transition occur? And what is the effect of the asymmetry in the rising and falling edges?

One of the worst contributors to data skew is the lack of alignment between the ongoing and the offgoing currents in the switching cells [37]. Chapter 1 showed that a skew of 1/64 of the clock period resulted in a SFDR of about 60 dB when generating

a sinusoid at one-sixty-fourth the clock frequency (see Figure 1.18). Figures 2.3 and 2.7 seem to indicate that BJT and CMOS current switches may have skews of 100 pico-seconds or higher, indicating that a straight binary-weighted DAC would suffer the same SFDR limitations at clock rates of well under 100MHz. Worse even than this basic switching asymmetry is that the data registers commonly used to drive the bit cells have an additional inverter delay between their complimentary outputs, doubling or tripling the transition skew.

Some of the best-performing DACs available have the top few bits segmented and thermometer-decoded in order to reduce the glitch magnitude [27, 52, 53, 54]. Although this is very effective, it comes at the cost of circuit complexity and higher digital feed-through.

One of the most recent designs [52] incorporates holding switches into each bit cell, as shown in Figure 2.13. Both of these switches are turned off briefly during the switching of the data registers in order to retime the transitions. This does two things: reduces cell-to-cell skew by moving the controlling clock edge closer to the current switches, and masks the asymmetry of the data register's complimentary outputs. The current transients seen at the output are completely determined by the current-switching devices, the current they switch, and the impedances driving them. The increase in dynamic range is believed to be 10dB.

One way of making the transitions linear is to turn all of the cells to a particular state (possibly all zeros) before applying the next valid code. This way the output node forgets the last data code before moving to the next. Unfortunately this demands that the DAC perform at least a partial settling twice as often as a normal DAC, reducing the maximum update rate. Output samplers perform the same function but the additional settling is that of the sampler, which can be faster than the DAC.

2.5.2 Digital feed-through and charge injection

As the amount of on-board switching is increased by the advent of decoding logic and retiming registers, more of it is seen at the output as digital feed-through noise. The major paths for these signals is through the substrate and between bonding pads and wires. The amount of switching is a nonlinear function of the input transition

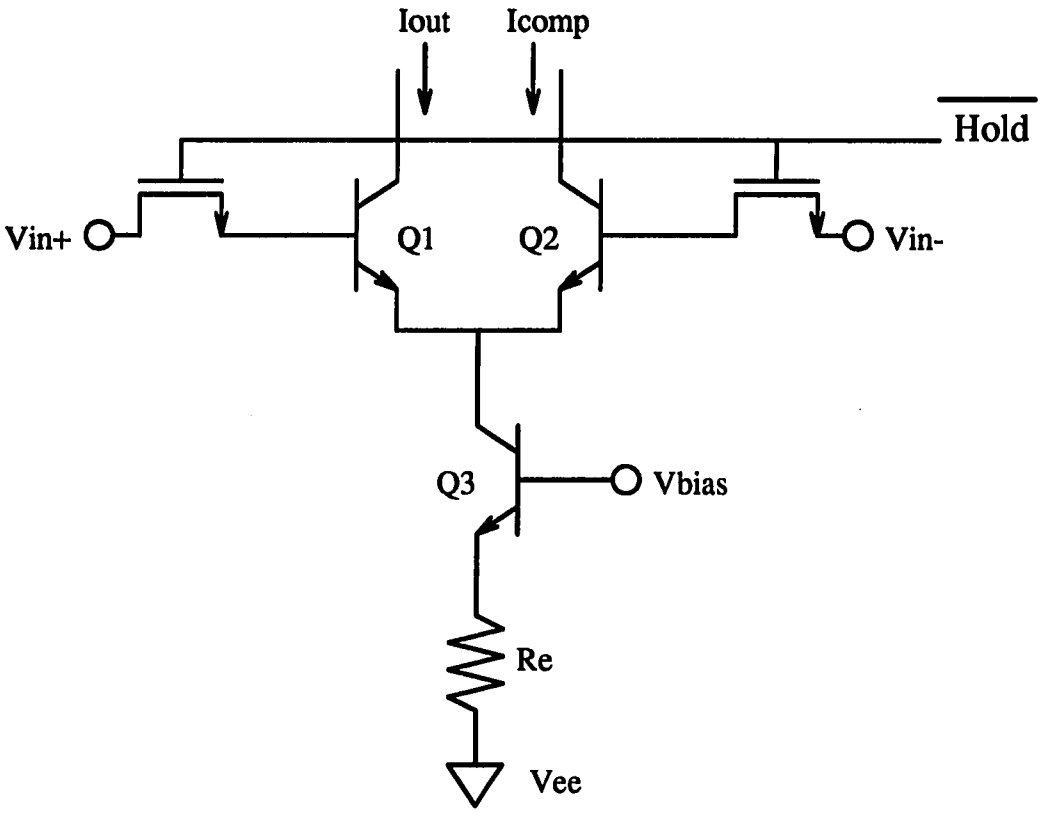

Figure 2.13: BiCMOS bit cell with sampling switches to reduce data skew

and therefore contributes to distortion. Unfortunately it is difficult to model the effects of both substrate currents and electromagnetic pickup.

The use of complementary Emitter Coupled Logic (ECL), with its lower signal swings and more uniform switching power transients, helps to reduce this digital noise. It is especially useful for bringing signals on-chip because the bonding wires and pads, which act as antennas to one another, see greatly reduced, complimentary signals.

Another source of nonlinear settling is the deviation in effective tail current caused by the charging and discharging of the depletion capacitors. This is shown in Figure 2.3 by the $I_{out} + I_{comp}$ term. The emitter bounce and current-source capacitance also cause fluctuations in this current.

2.5.3 Nonlinear capacitors and resistors

A nonlinear resistance in contact with the output node (where the current is passed through the load to generate the output voltage) can contribute a nonlinear term to the output current [27]. This is seldom a limitation in high speed DACs since this impedance is usually many orders of magnitude greater than the ideal load.

A more serious problem is the settling-time dependence on output level caused by nonlinear RC products. In both MOS and bipolar circuits, the output node sees a pn-junction diode and therefore a bias-dependent capacitance. Because this is the dominant distortion component in output-current samplers (and only a secondary effect in straight current-steering DACs), it is discussed more thoroughly in the sampling section at the end of this chapter.

2.6 Output Sampling

The best circuit technique to achieve spectral purity in a high-speed DAC is output sampling. This allows separate optimization of the transient and settled signals by using the sampler to mask the nonlinear settling inherent in a semiconductor switching network.

The previous section and Chapter 1 showed how only a fully-segmented and thermometer-decoded DAC could ensure that its transitions were linearly scaled to

the digital input step. Unfortunately, segmentation is very area-intensive and thermometer decoding beyond 5 or 6 bits is costly in logic and contributes heavily to power supply and substrate noise.

It is very difficult to construct a sample-and-hold or track-and-hold that is linear to more than about 80-90dB [25, 26, 55]. This is because these devices depend on inherently nonlinear devices, switches, which suffer from the speed/accuracy tradeoff outlined in the following paragraphs, and have the additional disadvantage of needing to hold a constant value on a capacitor during and after this nonlinear transition. Issues of droop, acquisition time, and acquisition slew also contribute to distortion and complicate the design. Many commercial devices are designed to interface with ADC and their linearity is guaranteed only in the hold mode, so transitions can be quite nonlinear.

The transitions of an output sampler (without the holding function) are determined by the settled input (DAC) current and the sampling devices. Because a sampler can be constructed from a relatively small number of transistors, its theoretical distortion figures can be both high and readily calculated. Unfortunately, as noted in Chapter 1, output sampling's advantages come at the cost of reduced signal amplitudes (due to pulse width narrowing), higher image amplitudes, and less circuit headroom.

Consider the Return-To-Zero (RTZ) sampler shown in Figure 2.14. This structure is very similar to the GaAs output device reported by Hsieh [56]. This differential scheme utilizes both the true (I_{out}) and complementary (I_{comp}) DAC currents (instead of steering the "off" current to ground). The sampling devices (M1-4) direct these currents to a dummy resistor (RL/2) when the DAC currents are changing, and to the outputs (V_{out} and V_{comp}) when they have settled. This means that while the DAC code is changing, the output floats up to ground.

The main error associated with this circuit is the settling time's dependence on the signal current. If a linearized approximation (similar to that used for the bipolar bit cell) is applied for the settling behavior of the sampler, a relationship can be found between the transient response and the output level. For the moment, charge-injection inaccuracies will be ignored.

The linearized short-circuit current-gain of a MOSFET can be found in much the

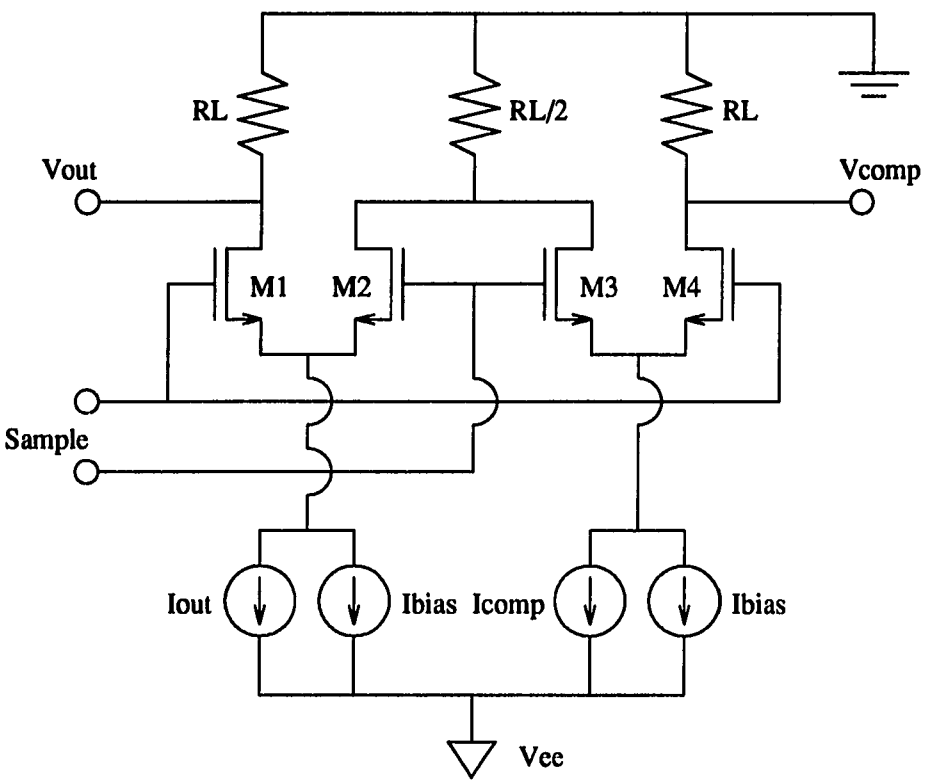

Figure 2.14: A simple return-to-zero sampling structure

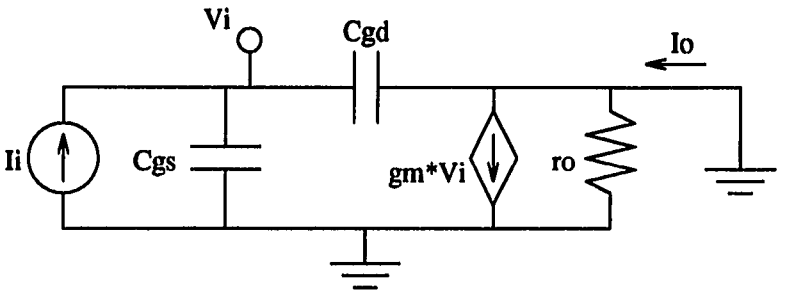

Figure 2.15: Small-signal diagram for calculating MOS f_t

same way as a that of a bipolar transistor (Figure 2.15). Since the output resistance is short-circuited, r_o can be ignored, and, realizing that $C_{gs} >> C_{gd}$, the current gain is simply:

$$\frac{I_o}{I_i} = \frac{gm}{sC_{gs}} \tag{2.19}$$

This implies that the DC gain of the device is infinite (a pole at the origin). Although this is not true, for this analysis - which ultimately is concerned with settling - the high-frequency characteristics of the MOSFET are modeled well, and the unity gain frequency is what is of interest:

$$f_t = \frac{gm}{2\pi C_{gs}} \tag{2.20}$$

In a MOSFET:

$$gm = \frac{dI_d}{dV_{gs}} = \frac{K'W}{L}(V_{gs} - V_t) = \sqrt{\frac{2K'WI}{L}} \tag{2.21}$$

creating a current-dependent time constant:

$$\tau_{sampler} = \frac{C_{gs}}{\sqrt{\frac{2K'WI}{L}}} \tag{2.22}$$

where, depending on the region of operation:

$$C_{gs} \approx WLC_{ox} \tag{2.23}$$

To determine the effect of this current dependence on the transfer characteristic, recognize that the linearity is related to the relative area of the output pulses. If the final settled value of a particular pulse is intended to be twice that of another, the area of the two pulses should be scaled by a factor of two as well. The area of a pulse is:

$$A = \int_0^T P\left(1 - e^{-t/\tau}\right)dt = P\left[T + \tau\left(e^{-T/\tau} - 1\right)\right] \tag{2.24}$$

where T is the update rate of the sampler, τ is the sampler time constant, and P is the pulse height. Note that as τ tends to zero, the area goes to PT, and as τ tends to infinity, the area goes to zero, as expected.

For the sake of simplicity, assume that τ is a function of the final settled value P, and not the instantaneous output level. The area of the pulse is now:

$$A = P\left[T + \tau(P)\left(e^{-T/\tau(P)} - 1\right)\right] \tag{2.25}$$

and a plot of this function versus DAC current, I, should approximate the transfer characteristic of the RTZ sampler for a given update rate, T. From the current dependence of equation 2.22 and realizing that $P = IR_L$ yields:

$$A = IR_L\left[T + \frac{C_{gs}}{\sqrt{\frac{2K'WI}{L}}}\left(exp\left[\frac{-T}{C_{gs}}\sqrt{\frac{2K'WI}{L}}\right] - 1\right)\right] \tag{2.26}$$

A plot of the nonlinearity of this equation is shown in Figure 2.16 for the circuit values listed in Table 2.3. Although this plot indicates a distortion level of only -64dB, this can be increased to -84db with the addition of a 50% standing-current bias in parallel with the DAC output. Another 14dB (to -98dB) can be gained by decreasing the update rate from 50 to 10MSPS, at the expense of system bandwidth. This approximation seems to agree well with experimental results [56].

In order to compare the performance of different sampler structures in different technologies, a figure of merit is needed. For the purposes of this work, power and area will be ignored but assumed to be reasonable (as in standing bias currents of less than the signal current). A possible figure of merit could be:

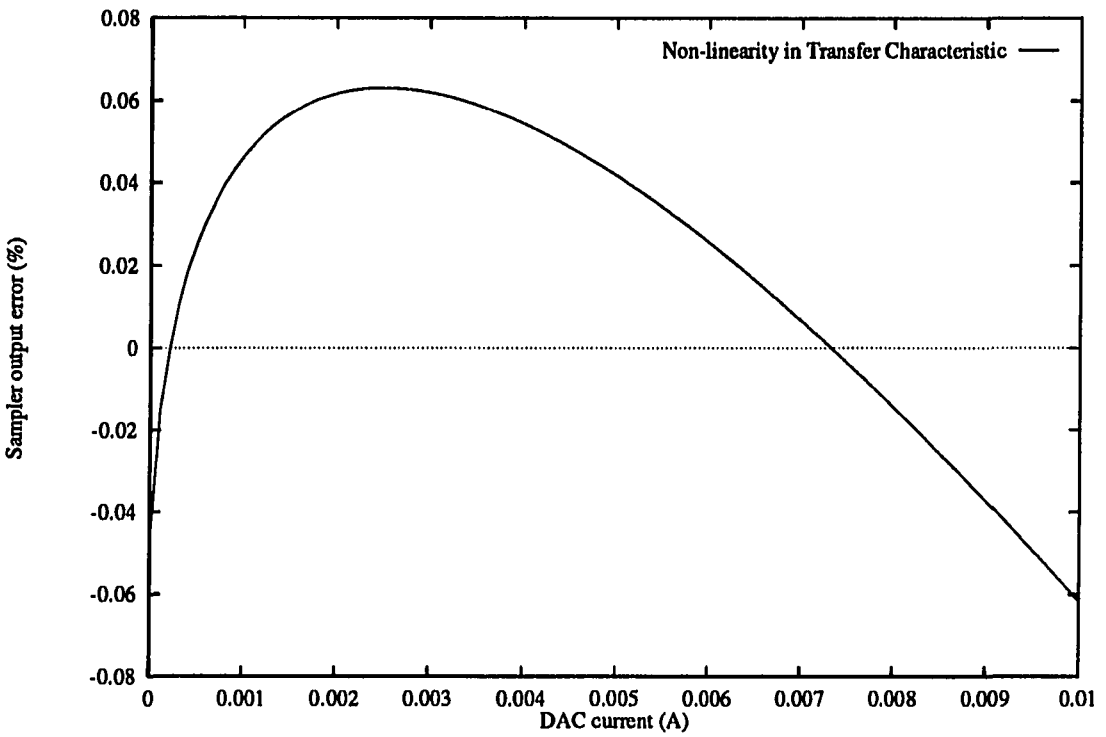

Figure 2.16: Estimated nonlinearity in transfer characteristic of an output sampler

Table 2.3: Sampler circuit values

Parameter	Value	Unit
R_L	100	Ohms
I_{max}	10	mA
T	20	ns

Parameter	Value	Unit
C_{gs}	2	pF
K'	30	uA/V^2
W	1333	uM
L	2	uM

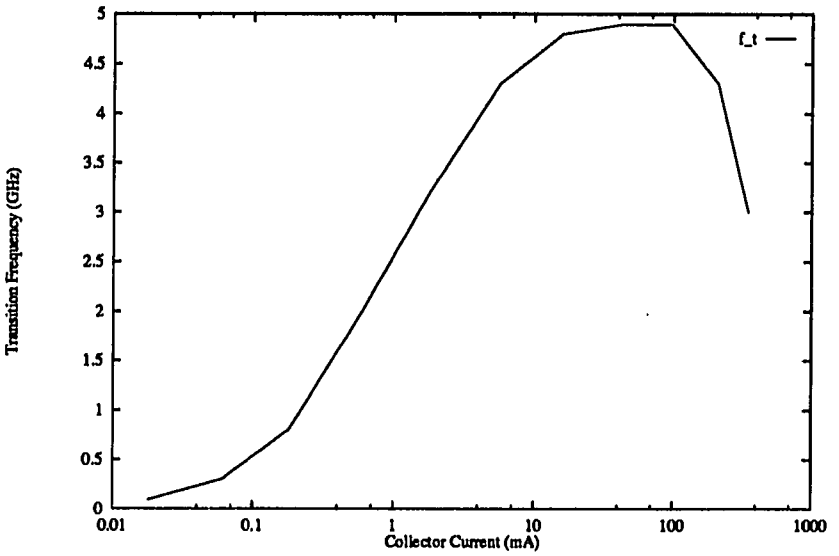

Figure 2.17: Simulated transition frequency for a bipolar junction transistor versus collector current

$$\eta = \frac{1}{\Lambda}\left[\frac{f_s}{f_t}\right] \tag{2.27}$$

where f_s is the sample rate, f_t is the transition frequency of the process, and Λ is the nonlinearity of the output. The above example would yield a figure of merit of:

$$\eta cmos = \frac{1}{1.26 \times 10^{-5}}\left(\frac{10\text{MSPS}}{1.5\text{GHz}}\right) = 496 = 53.9dB \tag{2.28}$$

Although bipolar processes will in general be faster, their f_t's have a more complex dependence on collector current (as can be seen in Figure 2.17 [33, p. 129]), leading to a higher nonlinearity for a given f_s/f_t ratio. To further complicate matters, the base current also varies nonlinearly with bias, and is very difficult to compensate for (Figure 2.18). Both of these figures use the same SPICE parameters given earlier in Table 2.1, but the size of the transistor has been scaled.

If equation 2.25 is applied to this process with a 50% bias current and full-scale current of 10mA, the figure of merit is:

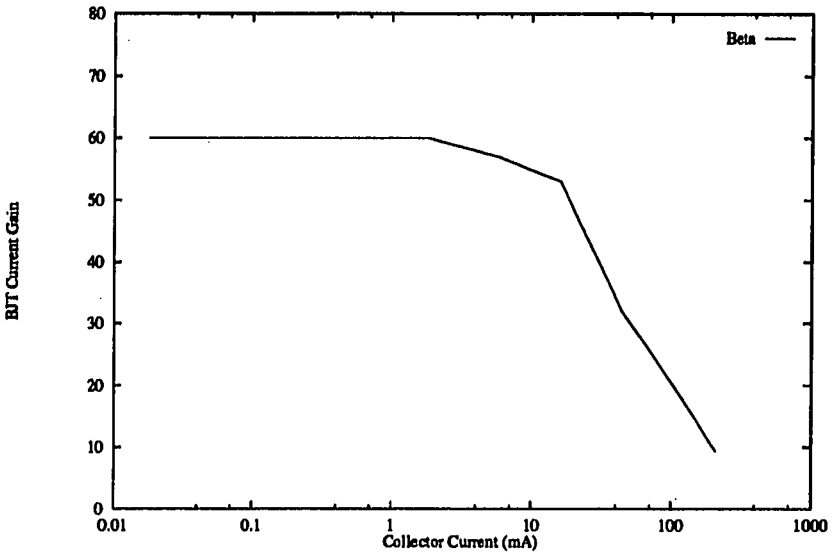

Figure 2.18: Simulated beta (current gain) for a bipolar junction transistor versus collector current

$$\eta_{bjt} = \frac{1}{8.2 \times 10^{-5}} \left(\frac{100\text{MSPS}}{5\text{GHz}} \right) = 243.9 = 47.7dB \tag{2.29}$$

prior to considering base current nonlinearities.

Remember that in both these cases all other sources of nonlinearity are ignored, including DAC errors. If this is the best the sampler can do, how can we expect much more from a standard DAC whose switching structure is much more complex and highly coupled?

One additional advantage can be gained by both samplers and sampling-and-holding structures. The errors that they both suffer from are more predictable than those of stand-alone DACs, so correction, calibration and predistortion techniques are easier to devise and apply. These ideas are discussed further in Chapter 3.

3. IMPROVED SPECTRAL PERFORMANCE THROUGH DIGITAL PREDISTORTION

This chapter demonstrates how the addition of digital signal processing and a feedback loop around the analog signal path can increase the spectral quality. This is done by altering the digital input signal based on a comparison of the ideal spectrum and that measured at the output of the DAC.

3.1 A Systems Approach

Chapters 1 and 2 showed how a DAC's spectral performance is determined by both its DC specifications (integral and differential nonlinearity, gain and offset errors) as well as its high-frequency, transient or settling characteristics. Both low- and high-frequency nonlinearities can produce new spectral components, harmonic and nonharmonic in nature. In a system trying to produce the highest possible frequencies, it is important to push the update rate of the clock as high as possible. When this is done it is very difficult to keep the nonlinear settling (or glitch) from becoming a serious problem.

Instead of correcting these errors, the current approach is to try to minimize them during the circuit design stage [28, 31, 37]. Unfortunately, minimizing all of the error sources at once generates conflicting design requirements. As an example, for glitch reduction as well as good DC matching (settled accuracy) it is desirable to segment the current sources and thermometer-decode the digital input. This drives up the digital circuitry and the number of switches, which leads to higher levels of digital feed-through, charge injection, and greater reference loop disturbance. The network also becomes larger and more distributed in nature, increasing parasitics, and complicating the settling and stability issues. Even output sampling was shown

to have serious distortion problems if the update rate is pushed to the edge of the process technology.

Another approach to the problem is to surround the converter with circuitry such that the system has higher spectral purity than the DAC. A feedback loop could be formed by evaluating the output signal of the DAC and using the resulting information to tune the system. An obvious area for investigation is some form of calibration, but most calibration schemes work only at very low update rates where the settled accuracy determines the performance. As the update rate approaches the settling time of the DAC, some form of dynamic calibration is needed.

Since most transition nonlinearities are a function of both the present and previous output level, if the effective spectral error for each possible step could be measured and stored, then perhaps this information could be used to output a different level which compensated for the nonlinear transient. The major problem with this strategy is that it is difficult to isolate and measure the frequency-domain errors of a single step. Other difficulties arise in storing the corrections for all possible steps ($\approx 2^{2N}$ for an N-bit converter), calculating and verifying the proper correction, and in compensating for temperature and aging effects.

Another technique, finding applications in Analog-to-Digital Converter (ADC) distortion compensation, is to pass the digital signal through a nonlinear filter to cancel converter nonlinearities [57]. The major source of distortion in many high-speed ADCs is due to slew-rate limiting in the front-end sample-and-hold, which leads to odd-order harmonics. Since the errors are well-known and well-behaved, they can be removed with digital post-processing. Unfortunately many DAC nonlinearities are not as easily mapped into a mathematical model. In fact, the transfer characteristics of even a good dynamic DAC can be changed drastically and unpredictably by minor changes in such parameters as DC offset, amplitude, and phase. This is because one of these minor changes may lead the signal through one of the more poorly-behaved transitions.

A variation on this technique has been applied to correct some of the more easily predicted DAC errors. Recently it has been demonstrated that a digital filter can be used to prewarp the DAC input signal to compensate for the sinx/x distortion [58, 59]. The drawback to this technique is that it assumes rectangular output pulses, which

are approximated well only by DACs whose output has settled for the majority of the clock period. This tends to limit the maximum update rate of the system. It is also an open-loop system with no method for checking the effectiveness of the compensation.

Since neither calibration nor open-loop nonlinear prefiltering will handle dynamic DAC errors, that leaves a closed-loop precompensation scheme. The idea is to find the digital input sequence which generates the output whose spectrum most closely resembles the desired. Although a group at Hewlett-Packard [60] has applied spectral evaluation and signal preconditioning to correct for linear-system magnitude and phase response, these ideas have never before been used for harmonic suppression.

Figure 3.1 shows the block diagram of an algorithm which will iteratively search for such an input (under certain conditions discussed below) [61]. A Fast Fourier Transform (FFT) is performed on the digital signal to determine the ideal spectrum. The Discrete Fourier Transform could be used for more generality, but the efficiency of the FFT makes it a better choice. The signal is then passed through the DAC, and the spectrum of the output is evaluated (in this case by resampling and another FFT). The FFTs are compared, and an error spectrum is generated which is then passed through an Inverse FFT (IFFT). The resulting digital time-domain signal is subtracted from the original input, creating a new, corrected signal. The process then repeats with the corrected signal until the error is within acceptable limits.

To better understand the algorithm, consider the following example. Assume the desired signal is a single-tone at a frequency ω. When the ideal digital sequence representing this tone passes through the DAC, it is amplified and phase-shifted, but remains at the original frequency (as if passed through a linear network). Figure 3.2 shows the vector representation of the input and output signal in the complex plane at the frequency ω. The difference between these two vectors is the error vector, which when subtracted from the original input signal results in the corrected signal. If the DAC amplifies and phase-shifts this signal in the same way it did the first input, the new output will lie very close to the ideal.

A more analytical example includes a second-order nonlinearity in the DAC transfer function described by the equation:

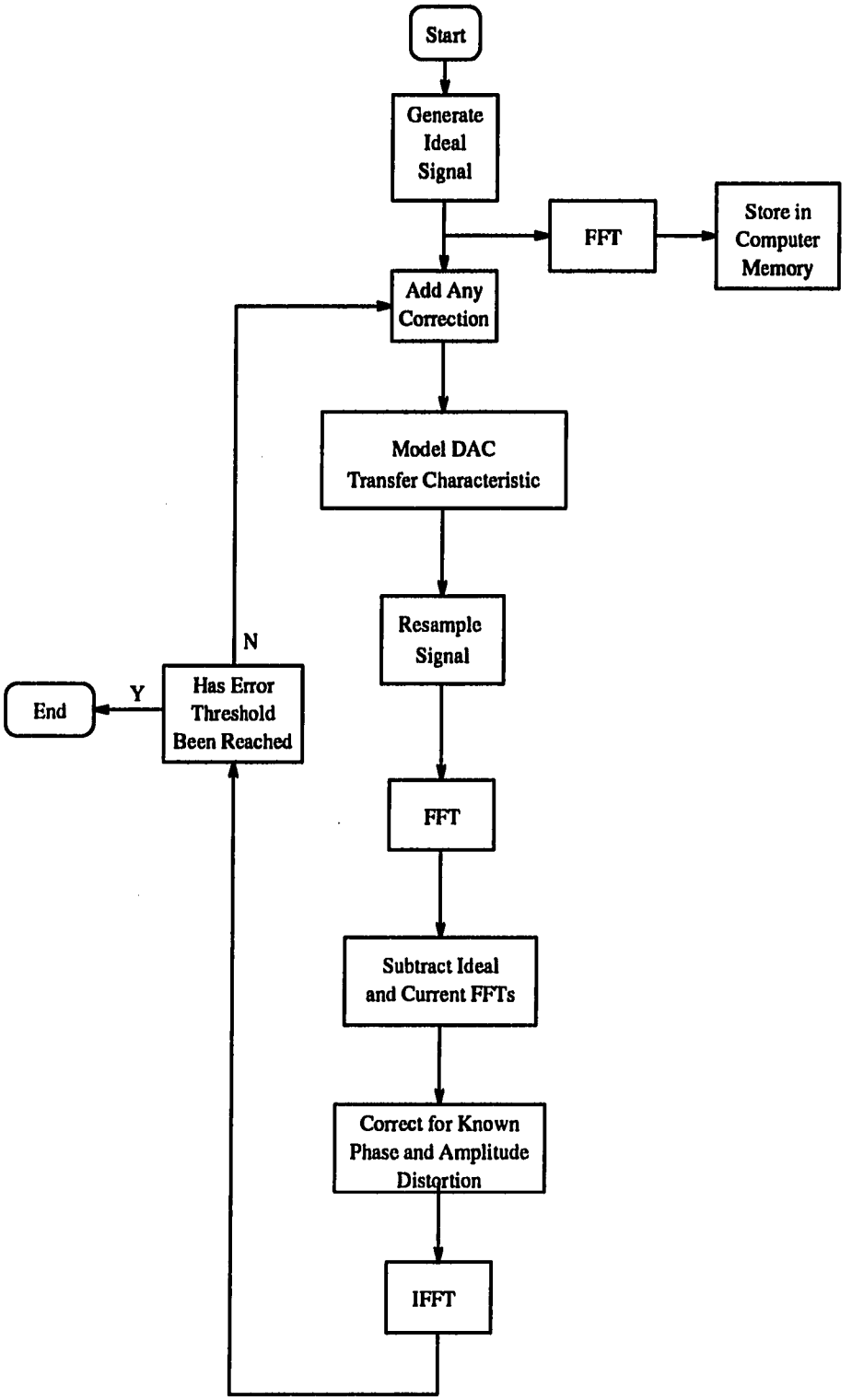

Figure 3.1: Possible algorithm for digital predistortion

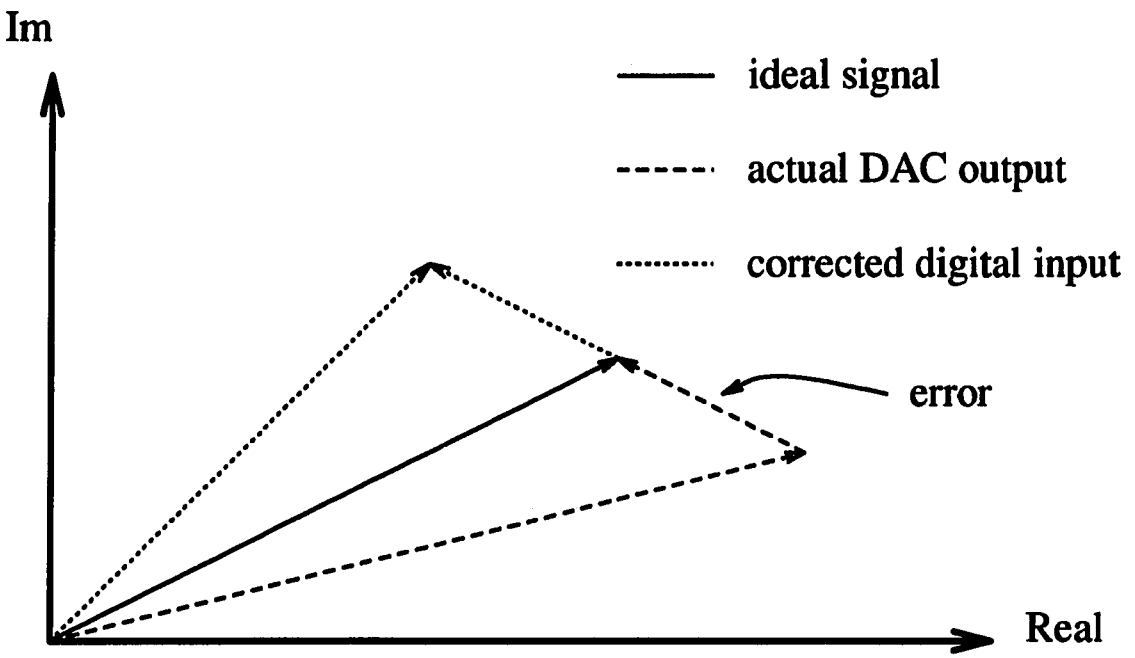

Figure 3.2: Complex-plane vector representation of the signals in a predistortion algorithm

$$y = x + 0.02x^2 \tag{3.1}$$

where y is the analog output and x is the digital input. If the input is a simple sine wave, then prior to correction the output spectrum will be ideal except for a second harmonic and DC offset, both 40dB below the level of the fundamental:

$$x(t) = cos(\omega t) \tag{3.2}$$

$$y(t) = cos(\omega t) + 0.02cos^2(\omega t) \tag{3.3}$$

$$y(t) = cos(\omega t) + 0.01cos(2\omega t) + 0.01 \tag{3.4}$$

On the first iteration, the algorithm will attempt to compensate for the distortion by subtracting the unwanted components from the digital input signal creating a corrected signal:

$$x'(t) = cos(\omega t) - 0.01cos(2\omega t) - 0.01 \tag{3.5}$$

When this new signal passes through the DAC, the output is determined by equation 3.1 as:

$$y'(t) = cos(\omega t) - 0.01cos(2\omega t) - 0.01 + 0.02[cos(\omega t) + 0.01cos(2\omega t) + 0.01]^2 \tag{3.6}$$

The first term in the bracketed quantity exactly cancels the two negative terms, resulting in a first-order correction. Note that the squaring of the input creates cross products between the fundamental and the correction signal (albeit drastically attenuated) that are similar to intermodulation distortion in multi-tone signals. These cross products will figure into the correction signal on the next iteration, assuming that they are of a significant amplitude.

The correction algorithm has two effects. The first is to reduce the magnitude of the spectral distortion, and the second is to push the majority of these components higher in frequency. If there is a limited bandwidth of interest (as there is in most

applications) the errors may soon be pushed out of it, but it is clear that as the signals and error sources become more complex, the algorithm will have to iterate many times to drive down the distortion sufficiently. It is also possible that the algorithm will not converge.

Although the above example is very simple, it demonstrates the basic technique and shows some of its advantages and limitations. The closed-loop nature of this technique should allow it to reduce all of the analog distortion components simultaneously, including sinx/x distortion (regardless of pulse shape), magnitude and phase distortion, as well as most DC and AC DAC nonlinearities. It can even remove perceived delays in the signal due to digital registers and analog time constants. What now must be determined is whether it will correct for the highly unpredictable errors of DNL and major-carry glitch.

3.2 The Predistortion Algorithm in Detail

Figure 3.3 is a block diagram of the hardware necessary to implement the predistortion algorithm, and shows strong similarities to the memory-based signal generation system of Figure 1.5. Note that the system is controlled by a microprocessor or other computation device that is also responsible for processing the FFT.

Although there are many possible predistortion algorithms, only one variation is evaluated through simulation and laboratory testing in the remainder of this work. The minor differences between the description that follows and the testing implementation are detailed in Chapter 4. On each iteration, this particular algorithm tries to remove all spectral errors that meet two criteria. Each spectral error in the DAC output, quantified by its magnitude and phase, is the difference between the desired signal at a particular frequency, and the actual measured output at that frequency. If the error magnitude is greater than the *error threshold*, and it is lower in frequency than the *bandwidth of interest*, the algorithm attempts to remove it. The bandwidth of interest can be specified either as a frequency or as a number of FFT bins.

The first step in the algorithm is to generate the frequency-domain representation of the desired signal. A continuous-time equation of the desired analog output is entered into the computer, which calculates the value of the equation at integer mul-

tiples of the clock period, T. This sampling of the desired signal generates a sequence with amplitude resolution equal to that of the computer. Simulations indicate that the noise floor induced by computational errors is below -150dB, so these errors will not be analyzed. For this discussion, the desired output signal is periodic with period equal to a binary multiple of the clock period (2^K, where K is an integer between 1 and a maximum limited by memory). The sequence is then passed through an FFT algorithm with 2^K points, resulting in the frequency-domain representation of the desired signal.

The next step is to quantize the time-domain sequence to the resolution of the DAC, and pass it repeatedly through the DAC, to generate the analog output signal. This output is resampled at the same clock period, T, and a new sequence results. By performing an FFT on this new sequence, a representation of the actual spectrum is obtained.

The error spectrum is the vectoral difference between the desired and the actual spectrums. To maintain stability and enhance convergence, system delays and phase shifts between the desired and resampled signals must be removed from the error spectrum before calculating the correction signal. To accomplish this, an estimate of the system phase characteristic is subtracted from the phase of the error spectrum. This is equivalent to sliding the two waveforms back and forth until they appear to "line up." The information necessary to do this is can be obtained by measuring the low-frequency phase shift of the system.

The correction signal is the result of the inverse FFT of the error spectrum, after removing all frequency components which are above the bandwidth of interest or below the magnitude error threshold. The correction is then subtracted (in the time domain) from the original sequence, creating a new input to be quantized and processed by the system. The algorithm iterates until there are no more error components fitting the correction criteria or a preset iteration limit is reached.

3.2.1 Signal restrictions

One restriction must be placed on the type of signal that this algorithm will handle efficiently. Many DAC errors, such as differential nonlinearity and major-carry glitch, contribute strong discontinuities and code sequence dependencies to the

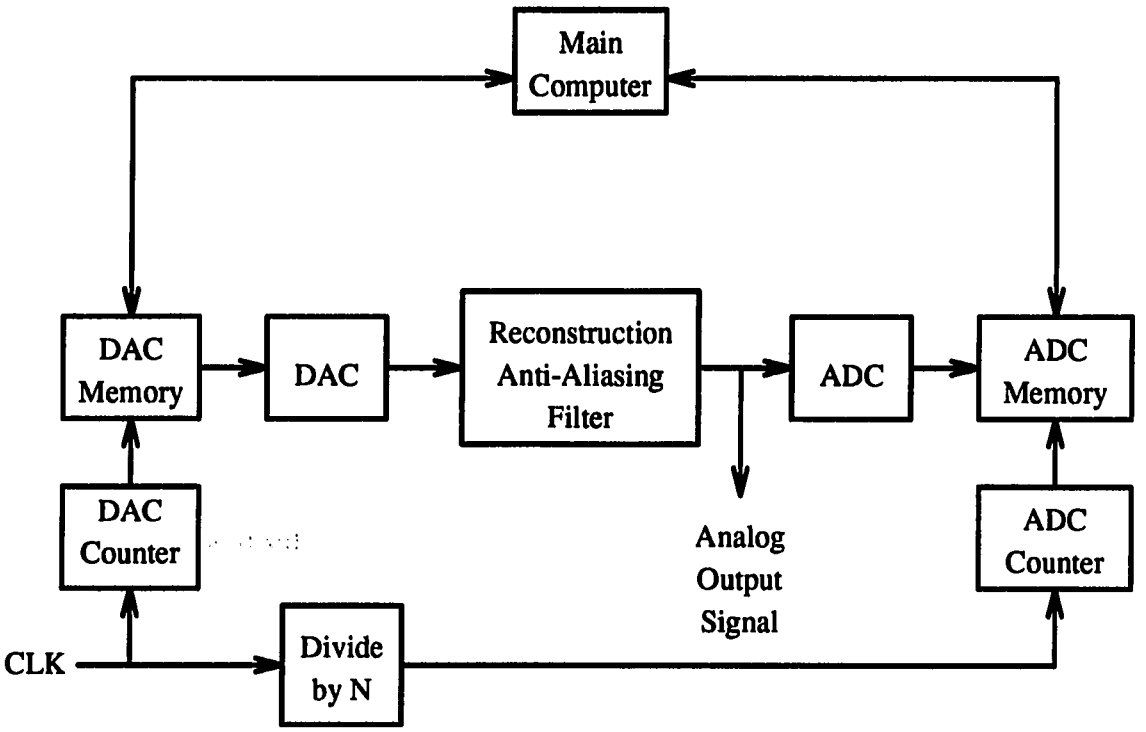

Figure 3.3: A hardware block diagram for implementing the predistortion algorithm

transfer characteristic. This means that even a slight change in the digital input code could result in a large spectral error, and thus the complete signal must be monitored. The algorithm will work best if the portion of the signal that the FFT evaluates includes the "Digital Repetition Period" (DRP) or time it takes for the digital signal to start running through the exact same code sequence. For a sine wave, this period will be the same as the analog output period only if the clock frequency is an integer multiple of the signal frequency. For more complex signals, the algorithm will work at maximum efficiency only on the portion of the signal evaluated by the FFT. Also, if the signal is aperiodic or not coherent with the clock signal, the FFT would have to be as long as the desired duration of those signals. The computation time of FFTs grows with the square of the number of points, so only a relatively small DRP can be handled.

It is also desirable to have the portion of the signal evaluated by the FFT (known as the "window") be exactly the length of the DRP, because the FFT algorithm evaluates the spectrum by assuming that the window is one period of an infinitely long signal. The spectrum is then the Fourier series representation of that window, with components only in specific frequency locations (or "bins") determined by one over the window length.

If the window is not the DRP, or an exact multiple, then the FFT is only an approximation to the spectrum. To enhance the approximation, a number of *windowing functions* (sometimes shortened to "windows") have been developed [62] which attempt to place the majority of the energy of the signal in the bins closest to the actual frequency.

The signal restriction of the DRP equaling the FFT window is only slightly more restrictive than that imposed by the length of memory in the predistortion system, especially if the FFT window length is a system variable. Fortunately, although long FFT windows are highly inefficient, they are necessary only for generating long DRPs, which are usually used for generating low-frequency signals with a high update-rate (i.e., fast clock). Chapter 2 demonstrates that these signals can usually be generated with higher spectral purity by decreasing the update-rate, and therefore a shortening of the DRP.

3.2.2 Stability and convergence

There are certain error sources that the algorithm cannot fully overcome. Errors due to quantization (the basic limit to the resolution of the DAC) cannot be corrected. Gaps in the transfer characteristic of the DAC that prevent it from spanning the full output range are also a problem, although simulation suggests that the effects of these errors may be reduced by the algorithm. These gaps are called positive DNL errors. Negative DNL errors - leading to nonmonotonicity - do not seem to limit the performance of predistortion. It is a fairly easy task to design a DAC that spans the full output range by using a radix less than 2 for the bit currents.

Large gain and phase errors in the converter can also compromise algorithm stability. As an example, consider a DAC gain error of two. When the ideal digital signal X is applied, the output is exactly twice that expected, or $2X$. The error signal is $2X - X = X$, and the corrected input is $X - X = 0$! It is obvious that the error on the next iteration will be X again, and the algorithm will oscillate between a signal of zero and $2X$. Phase errors of greater than 90 degrees will produce a similar instability because the algorithm is on the edge of increasing the errors instead of driving them down. Gain and phase errors that are not large enough to cause instability can still drastically increase the convergence time of the algorithm by limiting the effective correction at each iteration.

Both convergence and stability can be enhanced if the correction algorithm has a crude approximation to the frequency response of the converter system, and uses this information to adjust the correction signal. One way to evaluate the system's response is to apply a multi-tone signal (or frequency comb) to the system and measure the output spectrum. With the ADC and FFT hardware in place, this is simple to do to an accuracy sufficient to ensure stability.

The last serious threat to stability is interference from an outside source which is not coherent with the signal. The problem is that on different iterations of the algorithm the interference will have a different phase relationship to the signal, so a prewarping of the signal cannot cancel it. The problem is aggravated by the fact that the FFT (without a windowing function) will probably smear the interference over many bins. If this were not the case, the bins containing the interference could

be ignored. For this reason windowing is necessary in a hostile environment.

3.2.3 Subsampling and the ADC

For the algorithm to calculate an accurate correction signal, the analog-to-digital converter block has to capture the exact amplitude of the signal at precise points in time. This can be accomplished by a wide-band, low-distortion sample-and-hold amplifier (SHA) with low droop-rate (meaning that the held output should not change during the hold mode of operation). The SHA is followed by a high-resolution, low-distortion analog-to-digital converter. Since the signal being processed is periodic, the SHA can take a new sample every few periods, allowing the ADC a longer time in which to complete the conversion. This technique, called subsampling (or "beat" sampling, since it uses a beat frequency of the signal to do the sampling), drastically reduces the performance requirements of the ADC, and eases some of the specifications on the SHA.

To drive down errors due to system noise and clock jitter, many samples of the same point in the signal period can be taken and signal averaging techniques applied. The effectiveness of averaging depends heavily on the statistics of the noise and jitter sources, but experimental results demonstrate its value.

Averaging and subsampling do have their penalties. The amount of time needed to acquire the full signal is a function of both, and they require slightly more hardware and processing to implement.

3.2.4 Correction resolution

To explore the boundaries of the algorithm's spectral resolution, assume that it is the same as that of the digital signal applied to the DAC. This will of course ignore analog nonlinearities and any lack of superposition in the converter, but it will provide some idea of the theoretical limits of predistortion.

Every digital signal can be represented by the sum of impulses one-LSB high and separated by integer multiples of the clock period, T. If the signal has a fixed repetition rate, LT, its spectrum is the sum of the spectra of these impulses, which differ only in phase makeup.

The cosine series of a function that is symmetrical about zero and periodic on LT is:

$$f(t) = \sum_{n=0}^{\infty} A_n \cos(2\pi nt/LT) \tag{3.7}$$

where:

$$A_0 = 1/LT \int_0^{LT} f(t)dt \tag{3.8}$$

$$A_n = 2/LT \int_0^{LT} f(t)\cos(2\pi nt/LT)dt \tag{3.9}$$

The repetitive impulse every LT can be represented by:

$$f(t) = \sum_{k=-\infty}^{\infty} \alpha\delta(kLT + mT) \tag{3.10}$$

where α is the amplitude (one-LSB) and m is an integer between 0 and L-1 which represents the position of the impulse within a single period. This function's cosine coefficients are then:

$$A_0 = \frac{\alpha}{P} = \frac{\alpha}{LT} \qquad A_n = \frac{2\alpha}{P} = \frac{2\alpha}{LT} \tag{3.11}$$

If this function is shifted in time by mT, the coefficients are unchanged, but each cosine is phase-shifted by $2\pi mn/L$.

This impulse spectrum determines the smallest amount that a digital spectrum can be altered or adjusted. Of course, multiple impulses can be combined in such a way that certain frequency components cancel and others superimpose. The spectral amplitude α/LT, or $2^{-N}/L$ of full-scale for an N-bit DAC, defines a repetition-period-dependent limit to the dynamic range. This means that a highly oversampled signal has the potential to be more accurate than one with a repetition rate closer to the clock frequency. Unfortunately an adjustment of this magnitude cannot be made at a single frequency without affecting others. To take full advantage of this theoretical limit, the bandwidth that the algorithm corrects must be adjusted to one over the repetition rate.

For a given signal and clock frequency, the smaller the bandwidth that has to be monitored, the more efficient the algorithm becomes in both convergence and final dynamic range. As an example, consider a spectrum correct except for a small DC offset. If the bandwidth of interest extends from zero to just under $f_s/2$, the smallest DC adjustment possible without affecting other in-band frequencies is $\alpha/2$ (achieved by adding one-LSB to the signal at every other time point). If the bandwidth of interest is reduced to just under $f_s/3$, the smallest adjustment drops to $\alpha/3$ (by adding an LSB every third time point). The same correction resolution extends over any frequency band equal to one over the repetition period.

Although the function is nonlinear, a good approximation to the resolution is:

$$\approx \frac{\alpha f_i}{f_s} \tag{3.12}$$

where f_i is the bandwidth of interest. It should be noted that this implies that a perfect DAC generating an ideal sine wave will be limited by its quantization noise at a lower dynamic range than the same DAC being driven by an ideally predistorted digital sine. These conclusions are confirmed by the simulation, but unfortunately it is difficult to write an algorithm that takes full advantage of this limit. To do so would require a great deal of knowledge about the spectrums of all possible correction signals.

3.3 Variations of the Algorithm

3.3.1 Reducing hardware

The algorithm presented thus far is only one of the many possible. For example, why not find the error signal by a direct subtraction of the input to the DAC and the output of the ADC in the time domain, thus avoiding the FFTs? In some instances this may be a viable technique, but there are three drawbacks. One is that there is no inherent evaluation of the distortion magnitude in the frequency domain, and no clear way to determine when a certain dynamic range has been reached. The second problem is that there is no way to ignore certain frequency ranges, which cancels the advantages of being able to ignore interference and apply noise-shaping techniques. The third is simply that the magnitude of time-domain errors is not a good indication

of the distortion in the frequency-domain. However, one possible enhancement to the existing algorithm might be to use a combination of time- and frequency-domain subtraction to speed convergence.

Another way to avoid the ADC and FFT hardware and processing time is to do the spectral evaluation in the analog, continuous-time domain. An analog spectrum analyzer could be used, but the phase must also be acquired, and the information would still have to be digitized to correct the input signal.

3.3.2 Utilizing imaged signals

Although the frequency resolution of the algorithm is fixed by the available memory, there is no reason for the maximum frequency to be limited by the clock rate. The output spectrum of a DAC does not end at $f_S/2$ but rather repeats the same spectrum every f_S. Typically these images of the baseband signal are heavily attenuated by the sinx/x and settling time response of the DAC, but these effects can be corrected with predistortion if the spectrum can still be evaluated at the new, higher frequency.

Consider the block diagram of Figure 3.4, which differs from previous systems presented in this work in that the anti-imaging filter has been changed from a low-pass to a band-pass, and the SHA is used explicitly as a subsampler. The DAC generates a spectrum that repeats every f_S, the band-pass filter removes all but the portion between f_S and $3f_S/2$, and then the SHA and ADC convert the output back into the digital domain (Figure 3.5). If we assume for a moment that the sampler can accurately capture the signal at these frequencies, the output of the ADC is a baseband, or demodulated, version of the signal between f_S and $3f_S/2$. The predistortion algorithm can now be applied as before, yet the output signal's frequency has been increased. The limit to this technique is set by the sinx/x response of the DAC (dependent on pulse shape) and the bandwidth of the DAC settling characteristic (*not* the settling time which is many time constants).

This system has another advantage in that it can be used to minimize the number of bins in the FFT, drastically reducing the processing needs. The number of bins in an FFT is determined by the ratio of the highest frequency to the resolution required. As an example, if a 1MHz carrier is amplitude-modulated by a 10KHz

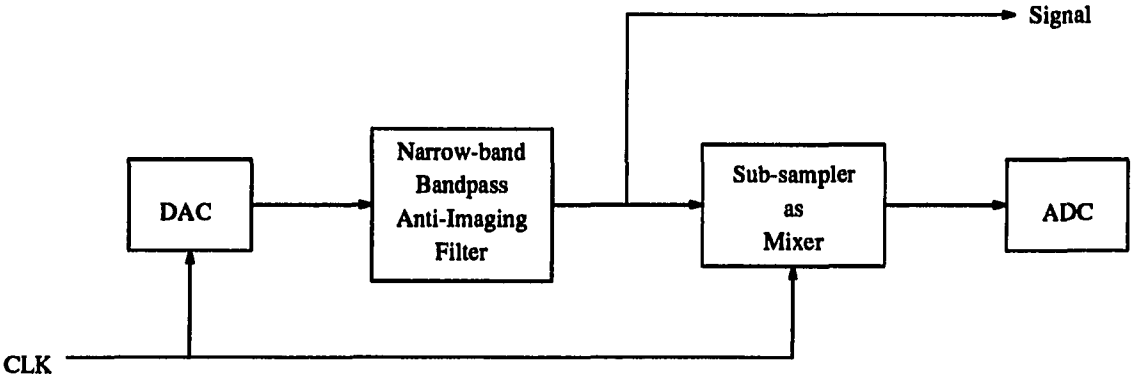

Figure 3.4: Block diagram of a system for utilizing imaged signals for high-frequency, narrow-band applications

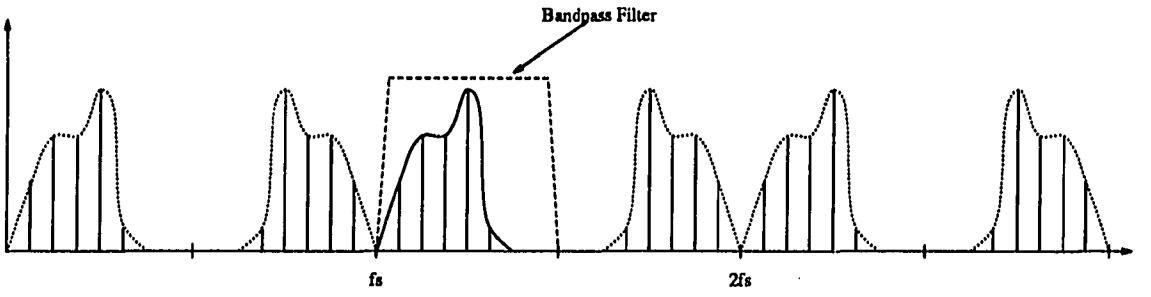

Figure 3.5: Spectrum of system utilizing imaged signals

tone, the minimum resolution and bandwidth required would be 10KHz and 1.01MHz respectively. This means that at least 101 bins would be needed (or 128 when rounded up to the nearest binary number necessary for FFT operation).

Now if the sampler is used to subsample, or demodulate, the DAC output signal, the resolution remains unchanged, but the maximum frequency is reduced. Using the above example and sampling at 900KHz would yield a baseband signal of 90KHz and 110KHz, as shown in Figure 3.6. The resolution is still 10KHz, but the maximum frequency is 110KHz, yielding a new FFT bin count of 11 (or 16 for the binary equivalent). Additional reduction is possible with careful selection of the subsampling frequency, and the minimum number of bins needed is set by the ratio of the signal bandwidth over the resolution required. Note that for the predistortion algorithm to work properly (by detecting and correcting all spectral components in a frequency range) the resolution of the system should be used and not that of the generated signal.

This new subsampling clock is usually generated by dividing down the DAC clock so that it is still coherent with the signal. In the case where imaged components are used, the two clocks can be the same.

The bandpass filter for this system must be very high frequency, but its magnitude and phase response can be corrected for by the algorithm, drastically easing the design. This filter acts as an anti-aliasing filter for the SHA and ADC, so its bandwidth cannot exceed $f_S/2$, but if it is narrower then the algorithm can be made to converge faster. The most troublesome part of this system is the SHA, which must have a bandwidth commensurate with the input signal while maintaining a high linearity.

3.3.3 Generating a wider class of signals

There are two options available if the frequency resolution of the signal generation system is not fine enough. The first is to increase or vary the memory length, but if the length is not a binary number, the FFT signal processing has to be changed to the more general, and less efficient, DFT.

The second option is to vary the master clock frequency by placing it in a phase-locked loop. This does not have the same drawbacks as the phase-lock based signal

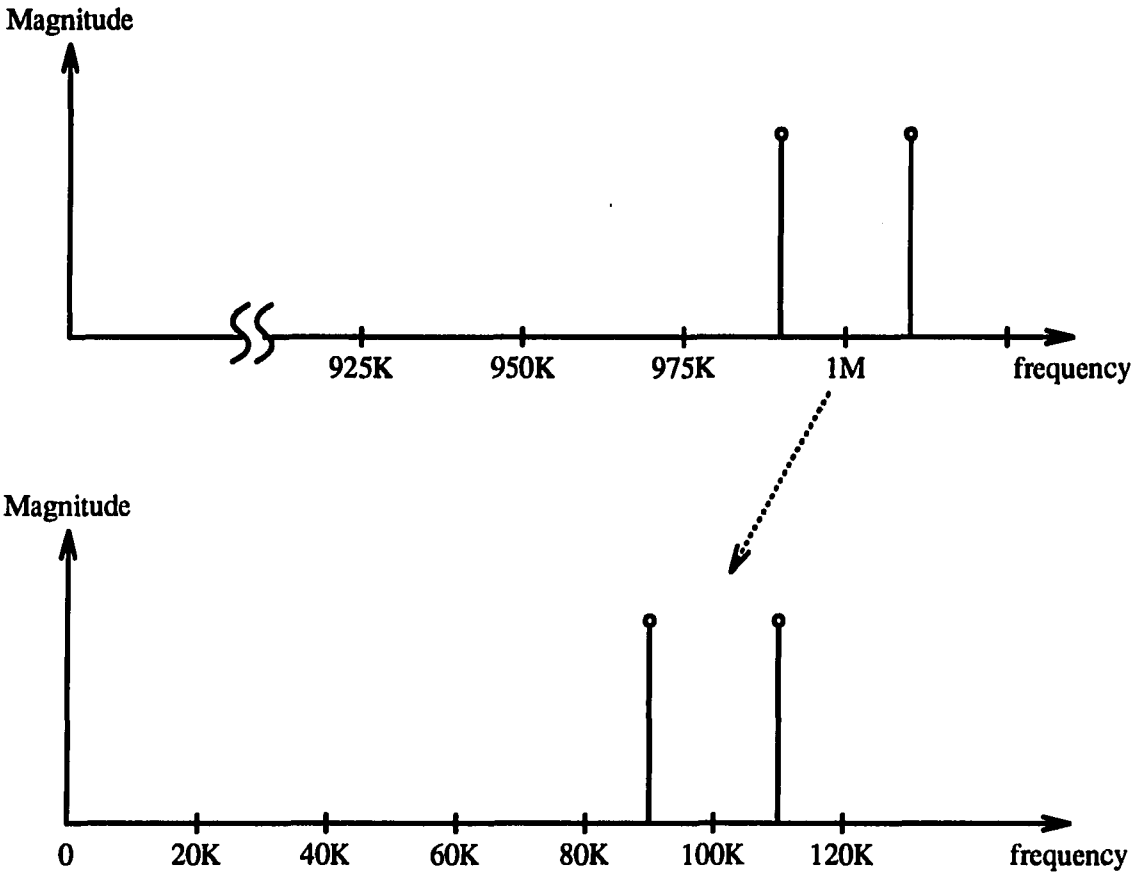

Figure 3.6: Spectrum of signals before and after demodulation by subsampling.

generators because the loop does not have to respond to transients since the clock, once set to the new value, does not change during signal generation.

3.4 Simulation Methods

To perform initial tests on the predistortion algorithm, a simulator was written in C, the flow of which is the same as that shown in Figure 3.1. This program went through numerous revisions to accommodate a wide variety of test conditions. Full data skew simulations, however, were deemed far too computationally intensive to combine with the predistortion algorithm. The algorithms ability to deal with data skew errors will be demonstrated through laboratory testing.

3.4.1 DAC modeling

Two basic modeling methods were used to simulate DAC nonlinearities. The first was a "deterministic" technique which used equations and simple logic operations to emulate the effects of integral and differential nonlinearity, quantization, gain, offset, and limited transition nonlinearities. Sinx/x distortion, a byproduct of sampling with a finite pulse width, was part of the signal without special modeling and depends only on the number of FFT points taken per clock period. The effects of the reconstruction filter were mimicked with a digital filter.

Although these deterministic simulations model many DAC error sources well, they do not capture the unpredictable nature of component mismatch (the DNL of these models is relatively simple, as is seen in the next section). To gauge the algorithm's ability to handle the significant code-to-code nonlinearity of large DNL, a transfer characteristic was generated with a random-number generator and a program that creates pseudo-Gaussian matching statistics.

3.4.2 Implementing the algorithm

The remainder of the simulator implements the algorithm in essentially the same way it is used in a laboratory on a real DAC. The program can generate numerous digital input signals to apply to the modeled DAC, the output of which is resampled by an ideal ADC with infinite resolution at as many points as are deemed necessary.

It should be remembered that if the reconstruction filter cuts off all of the signal above some frequency, the sampling rate of the ADC need only be twice that to completely capture the signal.

The program also contains FFT and IFFT routines capable of handling more than 2000 points, and the code necessary to generate the vector representation of the error between the ideal and actual spectrums. At each iteration of the program the SFDR and peak frequency-domain errors are calculated. These need not be the same if the peak error occurs in a bin containing part of the desired signal (e.g., gain error).

3.5 Simulation Results

It is impractical to include all of the results of the simulations performed to evaluate the predistortion algorithm. What appears here is an overview of the findings and a number of examples intended both to show typical performance and to illustrate any shortcomings. Throughout these results, times and frequencies are given relative to the clock since they have no real physical significance until they are applied to specific components.

Unless otherwise noted, the following defaults apply to the simulation model. The DAC update rate, from which all other times and frequencies are measured, is given the normalized value of $f_S = 1/T$. The DAC's resolution is 12 bits, its output range is ± 2 Volts, yielding an LSB size of 0.977 mV. The memory length (related to the maximum digital repetition period) is 1024. The algorithm's FFT defaults to 1024 points and a frequency resolution of $f_S/512$. This means the window length is 512T and that the generated signal is sampled twice per DAC clock period. This oversampling causes the sinx/x distortion to appear in the FFT output. The bandwidth of interest defaults to 154 FFT bins (or about $0.3f_S$), so the algorithm will only attempt to correct errors in this band. These defaults are summarized in Table 3.1.

The ideal DAC output can be computed from the following equation:

$$V_i(t) = K\left(\frac{\text{CODE} - 2^{(N-1)}}{2^N}\right) \tag{3.13}$$

Table 3.1: Simulation defaults

Section	Setting	Default Value	Comments
DAC	Update Rate	f_S	period: T
	Memory Length (L)	1024	
	Resolution	12 bits	
	Output Range	± 2 V	
Algorithm	FFT Window Length	512T	1024 points
	FFT Resolution	$f_S/512$	
	Bandwidth of Interest	$\approx 0.3 f_S$	154 bins

where N is the number of DAC bits, K serves as a scaling factor between the digital and analog domains, and CODE is the time-varying digital input signal. Note that the DAC produces -K/2 for the all zeros code, and K/2 for all ones. From the values in Table 3.1, the default DAC output is described by:

$$V_i(t) = 4\left(\frac{\text{CODE} - 2048}{4096}\right) \tag{3.14}$$

A brief summary of the simulation results appears in Table 3.2 and includes a description of the error type being simulated, the input signal (frequency and amplitude), the spurious-free dynamic range (SFDR) of the uncorrected spectrum, the number of predistortion iterations required to achieve a SFDR of 80 dB, and the resulting final SFDR. In many cases the time-domain waveforms are not shown due to the difficulty in visually distinguishing small errors on top of large signals. Where the time domain is of interest either the errors are plotted separately or are greatly exaggerated.

As was mentioned earlier, full data skew simulations were deemed too computationally intensive to combine with the predistortion algorithm. This is due to the fact that to accurately represent skew errors, the number of points per DAC clock would have to be on the order of the ratio of the skew time to the clock period (if not higher). This pushes the size of the FFT very, very high. The algorithm would then have to iterate on this size signal in the frequency domain, and since the FFT solution time is related to the square of the size of the FFT, this is clearly impractical.

Table 3.2: Summary of predistortion simulation results

Error Type	Comments	Input Freq. @ Amp.	Iter. No.	Largest Initial SFDR	Largest Final SFDR
INL	$2^{nd}, 3^{rd}, 5^{th}$	$3f_s/512$ @ 1	11	26.0dB	81.1dB
INL	bw = 0.03	$3f_s/512$ @ 1	6	26.0dB	89.4dB
DNL	Monotonic	$3f_s/512$ @ 1	N/A	28.0dB	34.8dB
DNL	Nonmonotonic	$3f_s/512$ @ 1	11	28.0dB	83.2dB
NonLinear Settling	Second Order	$3f_s/512$ @ 1	13	22.0dB	80.2dB
Combined Errors	Multi-Tone Signal	$3f_s/512$@0.5 $7f_s/512$ @ 0.25 $12f_s/512$ @ 0.125	16	32.2dB	81.1dB
Random	ENOB = 11	$3f_s/512$ @ 1	12	69.0dB	82.8dB

3.5.1 Integral nonlinearity

The DAC transfer characteristic shown in Figure 3.7 is created by passing the ideal DAC output (described by equation 3.14) through the warping function:

$$V_o = A + (1+B)V_i(t) + CV_i^2(t) + DV_i^3(t) + EV_i^4(t) + FV_i^5(t)... \qquad (3.15)$$

where the coefficients are:

- DC offset - A=0.05
- Gain Error - B=0.08
- 2nd Order - C=0.1
- 3rd Order - D=0.01
- 4th Order - E=0.0
- 5th Order - F=0.04

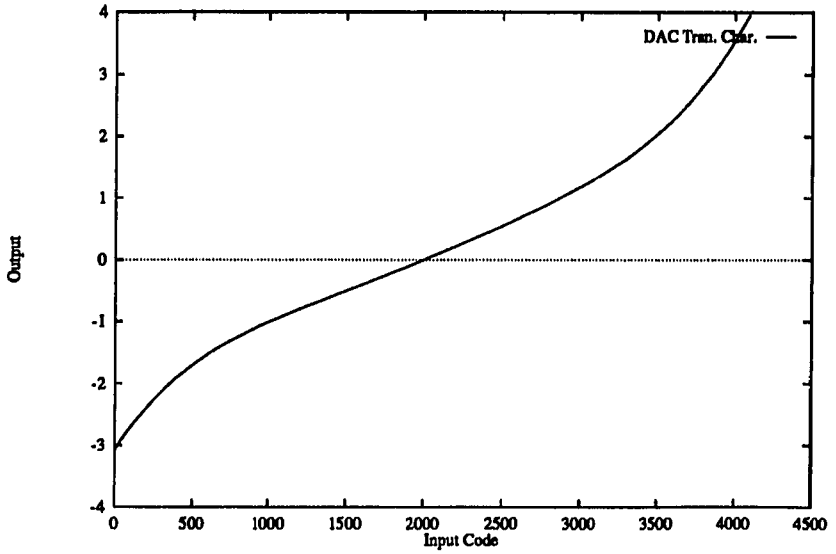

Figure 3.7: DAC transfer characteristic for INL simulations

- All Higher Orders are zero

A 1 Volt peak-to-peak sine wave at $3f_S/512$ passing through this DAC model produces the spectrum shown in Figure 3.8. The spectral errors are dominated by a DC component (-20.0 dB) and the second harmonic, and yield a SFDR of only 26.0 dB. After 11 iterations the predistortion algorithm increased the SFDR to 80.4 dB. When the same simulation was repeated for a bandwidth of interest of $0.03f_S$ (one tenth the default) the algorithm was able to achieve 89.4 dB (69.4 dBc) in only 6 iterations. Figure 3.9 shows a plot of SFDR versus iteration for the $0.3f_S$ bandwidth case.

3.5.2 Differential nonlinearity

It has been mentioned that the predistortion algorithm has trouble correcting the spectrums of DACs whose outputs have large regions in the output range that cannot be reached. This is caused by large, *positive* DNL. Negative DNL, on the other hand, indicates that there is a lack of monotonicity, or that the DAC covers the same range multiple times.

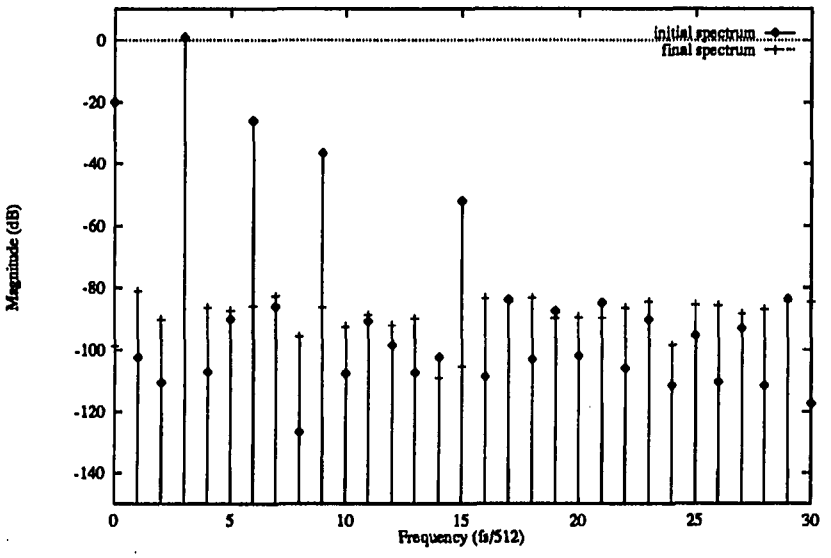

Figure 3.8: Predistortion algorithm's simulated response to integral nonlinearity errors

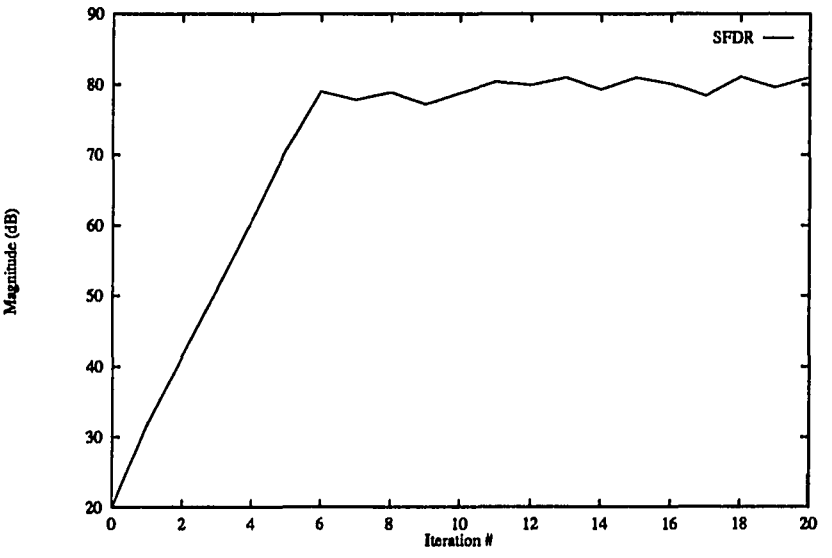

Figure 3.9: Predistortion algorithm's simulated SFDR versus iteration

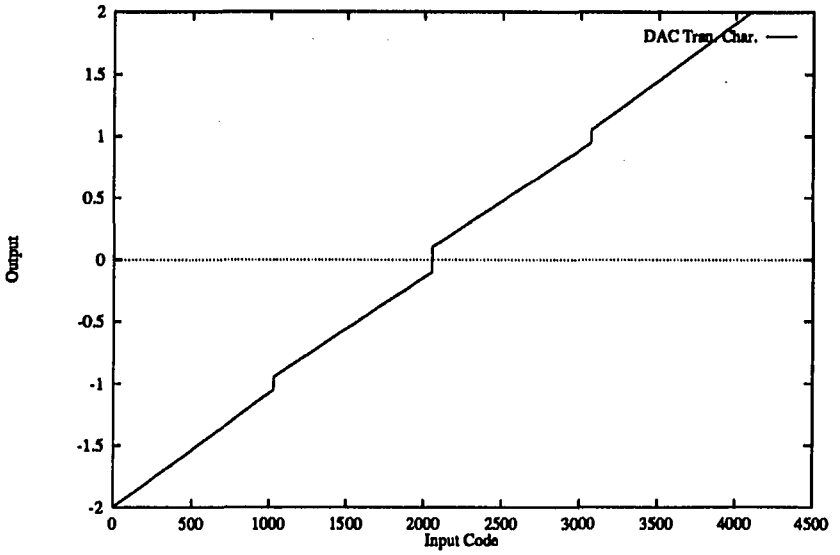

Figure 3.10: DAC transfer characteristic with positive DNL

To demonstrate the algorithms performance on DNL, a DAC model was constructed which is piecewise linear and contains discontinuities. Figure 3.10 shows the DAC transfer characteristic for positive DNL of 0.2 V or 201 LSBs which contains discontinuities placed to resemble the first two major carry locations. The initial spectrum, shown in Figure 3.11, yields a SFDR of 28.0 dB. Although the algorithm was never able to achieve 80 dB of SFDR (hence no entry in the iterations column), it did improve by almost 7 dB. Figures 3.12 and 3.13 show the transfer characteristic and spectrums for the case of a negative DNL of 201 LSBs. In this case predistortion was able to improve the SFDR from 28.0 to 83.2 dB in 11 iterations. The before and after time-domain signals are shown in Figure 3.14.

3.5.3 Other simulations

The effect of non-linear settling was investigated by altering the step response of the DAC model. A linear first order low-pass step response takes the form:

$$h(t) = g[1 - \exp(-t/\tau)] \tag{3.16}$$

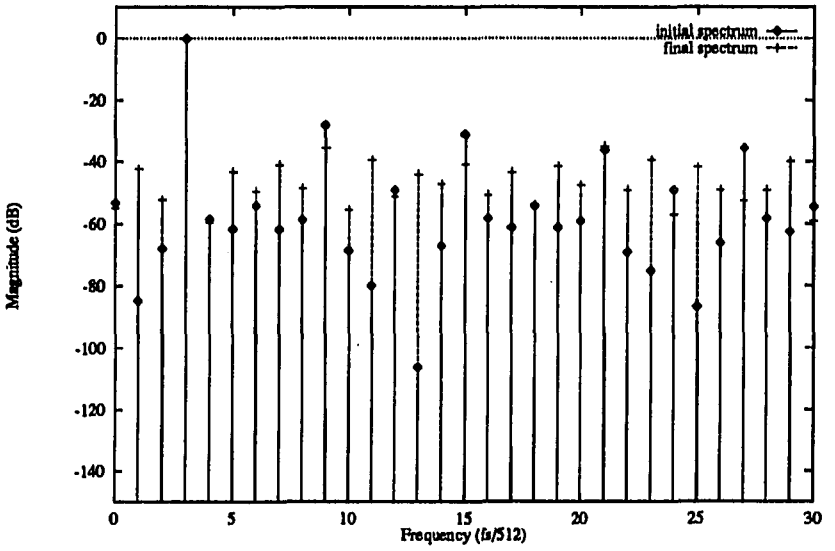

Figure 3.11: Simulated spectral response to positive DNL errors

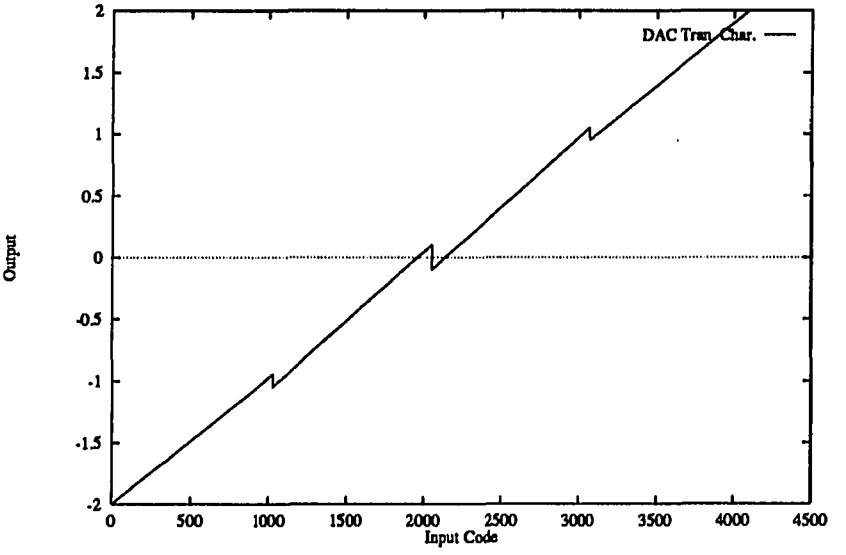

Figure 3.12: DAC transfer characteristic with negative DNL

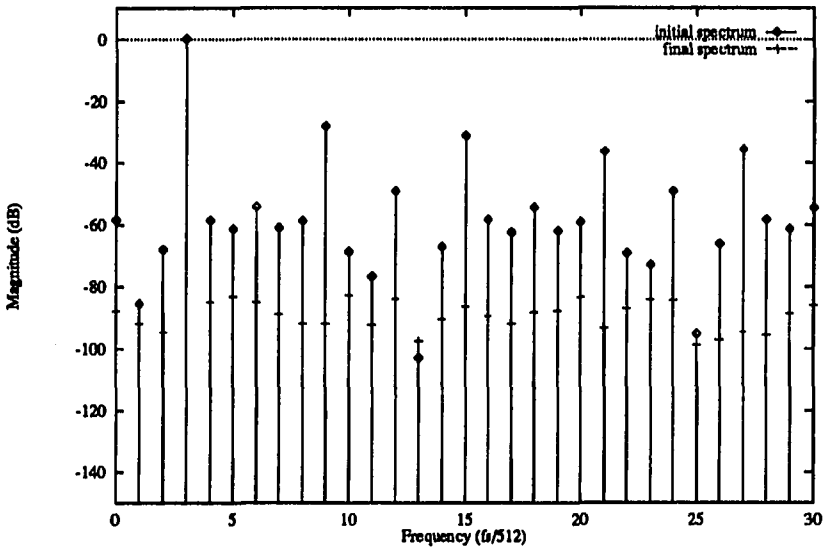

Figure 3.13: Simulated spectral response to negative DNL errors

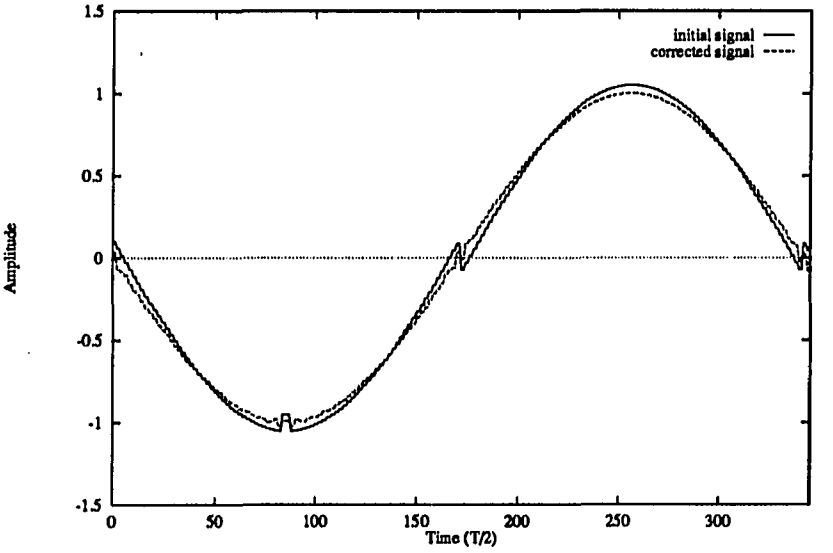

Figure 3.14: Simulated time-domain signals for a DAC with negative DNL

where g is a gain and τ is a constant determined by the system. If τ becomes a function of the input (V_i as described by Equation 3.14) then the settling is nonlinear. For the simulation described in Table 3.2, the time constant is a function of V_i^2. Predistortion was able to increase the SFDR under this condition from 22.0 to 80.2 dB in 13 iterations.

Another simulation, described as combined errors in Table 3.2, shows how the algorithm reacts to a multi-tone signal in the presence of multiple error sources. The input is the sum of three sine waves at different frequencies and amplitudes. The DAC model is a combination of INL, DNL, and nonlinear settling as described in the preceding paragraphs, but of lower magnitudes. The initial spectrum resulted in a SFDR of 32.2 dB, and after 16 iterations this was improved to 81.1 dB. As predicted earlier in this chapter, more iterations are necessary as the number of input signal components and error sources increases. This is because more mixing (or intermodulation) products are produced.

To evaluate the algorithms performance in the face of unpredictable code position, a gaussian pseudo-random number generator was used to create a realistic DAC transfer characteristic. The statistics of the number generator were adjusted until the DAC provided about 11 effective bits. The effective number of bits (ENOB) of a converter can be found from a measure of dynamic range by solving Equation 1.42 for N. Typically SNR is used, but in this case it is approximated by SFDR. This is valid because we are only concerned with relative quantities. Although the effective number of bits is somewhat signal dependent, it gives a frame of reference. The 69.0 dB of the last entry in Table 3.2 yields about 11.2 effective bits. The predistortion algorithm, after 12 iterations, was able to increase these numbers to 82.8 dB, or almost 13.5 effective bits. Note that the number of effective bits has passed the 12 bit resolution of the DAC. This is made possible by the algorithm "shaping" the spectrum. This is similar to delta-sigma converters in that the unwanted spectral components have been moved out of the bandwidth of interest.

3.6 Advantages and Limitations of Predistortion

These simulations have clearly shown that the predistortion algorithm can drastically improve the spectral response of a DDS signal generation system by compensating for a wide range of DAC and filter errors. Gain, offset, and low frequency INL and DNL are no longer limiting factors in these systems because DACs can easily be designed with a bias away from positive, and towards negative DNL errors. The resulting DAC can be coupled with the algorithm to remove the remaining DC nonidealities. If all frequency dependent distortion components react similarly, then the algorithms practical limit should be set not by the DAC and filter but by the acquisition circuitry.

Probably of most interest is the fact that the dynamic range of the DAC modeled in these simulations can be extended beyond that predicted by its number of bits. With the exception of the positive DNL case, all simulations were able to achieve 80 dB SFDR, closer to that expected in a 13 bit DAC. This is made possible by the spectral shaping of the algorithm, and further simulations indicate that this is close to the limit achievable by a 12 bit DAC utilizing a $f_s/3$ bandwidth of interest.

In addition to eliminating the need for sinx/x correction, predistortion also lowers the demands on the reconstruction filter and allows for higher DAC update rates without compromising performance. This is because the magnitude response warping usually associated with pushing the bandwidth is easily equalized by the algorithm. However, simulations indicate that it would be difficult to extend the update rate of a DAC much more than a factor of two or three beyond the traditional rate of one over the 1/2 LSB settling time.

3.6.1 Hardware overhead

Unfortunately, these advantages come at the cost of significant hardware overhead because every new signal must be run through the algorithm. The two largest contributors are the FFT computation circuitry and the Analog-to-Digital Coverter (ADC) which resamples the analog output. The ADC's accuracy has to be commensurate with that desired by the system, or the loop will converge to the signal that compensates for transfer characteristic errors in both the ADC and the DAC.

To put it another way, if we assume that the DAC and ADC transfer errors are commutative, it is impossible to determine which component contributed the error without independent knowledge of the intermediate (analog) signal. This means that the algorithm will attempt to correct the combined transfer function, possibly resulting in *inverse* ADC transfer errors appearing in the DAC output. For example, an INL bow in the ADC due to a partial squaring of its input will manifest itself, after predistortion, as a square root bow in the DAC output.

An ADC that meets these needs would be a more challenging design task than the DAC if it were not for the fact that by proceeding the ADC with a fast, accurate SHA, the bandwidth and update rate requirements of the ADC are lowered. This is accomplished by using the SHA as a subsampler, as was discussed earlier in this chapter. A new tradeoff is thus written between the speed of the ADC and the hold time of the SHA.

Some form of analog spectral analysis might reduce the component count and provide correction information more quickly. Accuracy bootstrapping, which uses an iterative approach to improving the accuracy of both the ADC and DAC [63], may ease the ADC design issues. It may also be possible to separate the DAC and ADC errors so that only those in the DAC are corrected. As the algorithm now stands, the DAC spectrum generated with predistortion is limited by the spectral purity of the SHA and ADC.

3.6.2 Signal and system restrictions

Another drawback to digital predistortion is that it works at maximum efficiency only on a limited class of signals — those with relatively small digital repetition periods or of short duration. If digital predistortion techniques can be extended effectively to a wider class of signals, its viability in test and signal-generation systems would be greatly increased. This could be done by extending the effective FFT window or by generalizing the conclusions drawn from observations of limited duration. Or, with the design of a sliding FFT window, it may be possible to operate in real time (with a certain latency).

Digital predistortion, as it has been presented here, is not readily compatible with many existing DDS systems for a number of reasons, not the least of which is

its inability to operate in real time. It is possible that this limitation can be eased somewhat by the storing of a large variety of possible signals. For example, a two frequency FSK system could store spectrally pure versions of the two frequencies with various phase relationships, and an algorithm could be designed to minimize the resulting phase transitions. Various other options may exist to meet particular system needs.

4. TEST SETUP AND EXPERIMENTAL RESULTS

4.1 Overview

In order to verify the concepts behind digital predistortion, two test boards (called "DAC" and "ADC") were designed and fabricated to apply the algorithm to a commercial DAC. System control, FFT processing, and spectral comparison were implemented on a Hewlett-Packard HP9000 Series 425T Workstation (hereafter referred to as the "workstation") connected to the test boards by a Motorola MC68HC11 Micro-Controller Board, which acted as a serial-to-parallel interface.

A simplified schematic of the test system appears in Figure 4.1. The test system design was partitioned into four sections: generation circuitry, acquisition circuitry, clocking circuitry, and instrumentation. The individual design considerations are covered separately in this chapter, but in all cases, flexibility and reliability are the main issues. Both boards use jumpers and dip switches liberally where performance would not be jeopardized, to allow system reconfiguration. There are also multiple input/output ports (I/O) where the signals, both analog and digital, can be evaluated or re-routed to external components (e.g., dedicated FFT hardware or another DAC board). A small prototype area for adding additional circuitry was also allocated to each board.

The test setup has four modes: memory load, free run, acquire, and memory read. During memory-load mode, the workstation (operating through the 68HC11) fills the DAC memory with the signal to be generated by controlling both the address and data buses. During this mode all other circuitry is quiet.

In free-run mode, the workstation is removed from both the buses, the ADC board is quiet, and the DAC memory is sequenced by its address counter (providing steadily increasing addresses, and recommencing at zero after rolling over). The

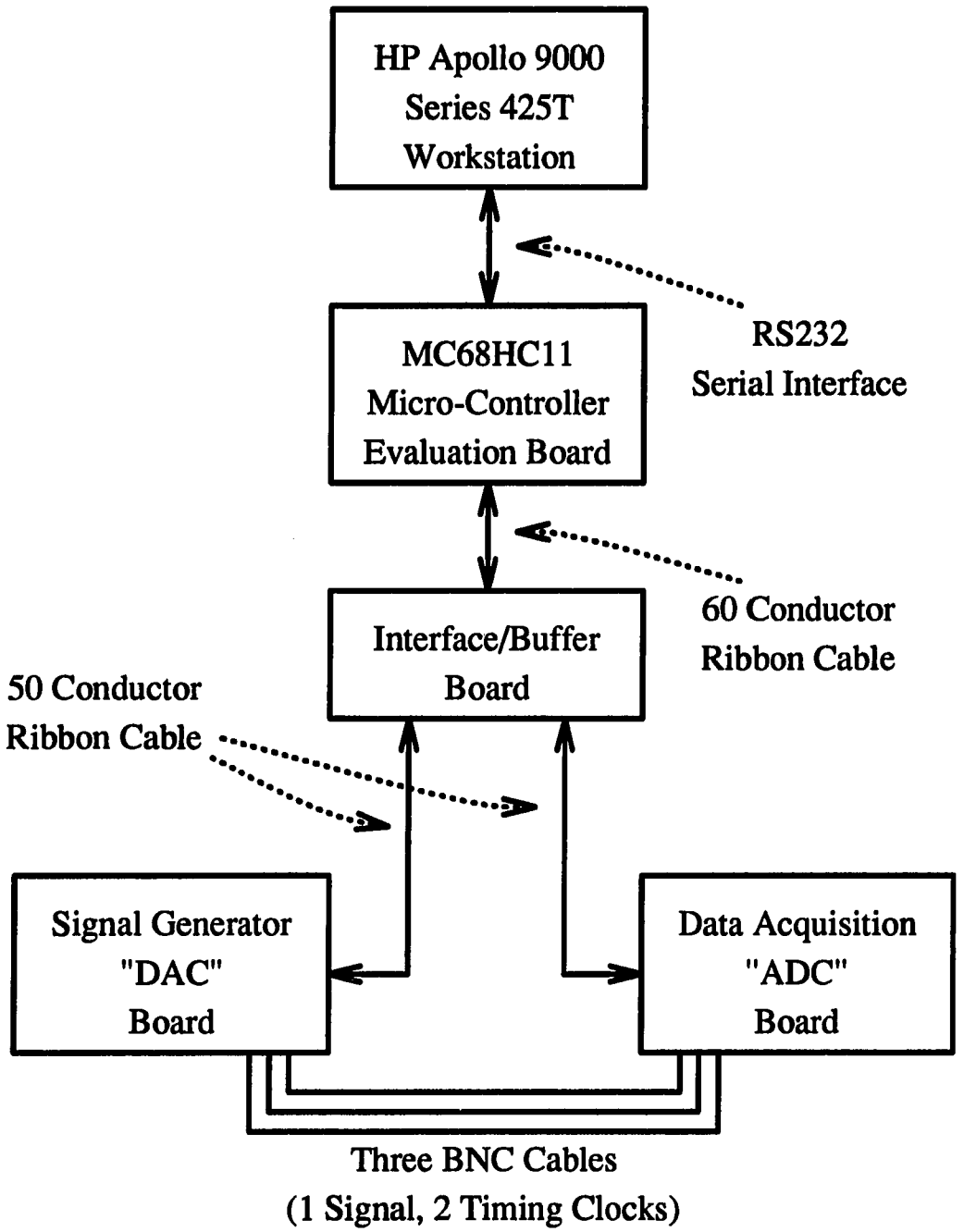

Figure 4.1: Simplified test setup schematic

output data from the memory is used to drive the DAC, which provides an analog output signal.

Acquisition mode is similar to run mode in that the workstation is inactive and the DAC is providing an output, but now the sampling of that output is performed by a sample-and-hold and analog-to-digital converter combination. The output data is loaded into the ADC memory whose address is controlled by a counter in much the same way as the DAC memory.

Once the ADC has captured a full period of the DAC signal (or multiple periods when using signal averaging), the system enters read mode, where the workstation again controls the address bus and receives data from the ADC memory. During this phase the DAC board is quiet. After the workstation has computed a correction signal the system is ready to return to load mode.

4.2 Generation Circuitry

Figure 4.2 is a simplified schematic diagram of the DAC and ADC boards. The DAC board contains the circuitry needed to receive the digital signal from the workstation and transform it into an analog, continuous-time signal. Most of this circuitry operates at the fastest clock rates in the system. Figure 4.3 is a detailed schematic of the DAC, optional output buffer (the AD845), memory components and bus interface circuitry.

4.2.1 Digital-to-analog converter

At the heart of the generation circuitry is the Analog Devices AD568 DAC. This commercial component was chosen not just for its speed (0.025% settling in 35 ns) and resolution (12 bits), but also because there are no on-chip registers for the digital data. Such registers can ultimately limit the update rate and also remove any controllability over data skew (making testing of the algorithm more difficult).

The main reason for choosing the AD568 is that its spurious-free dynamic range is limited by dynamic errors (especially data skew). This makes it an ideal component on which to evaluate the predistortion algorithm. The DC linearity is better than 1/2 an LSB (achieved by an R-2R ladder, segmentation, and laser trimming of thin-film

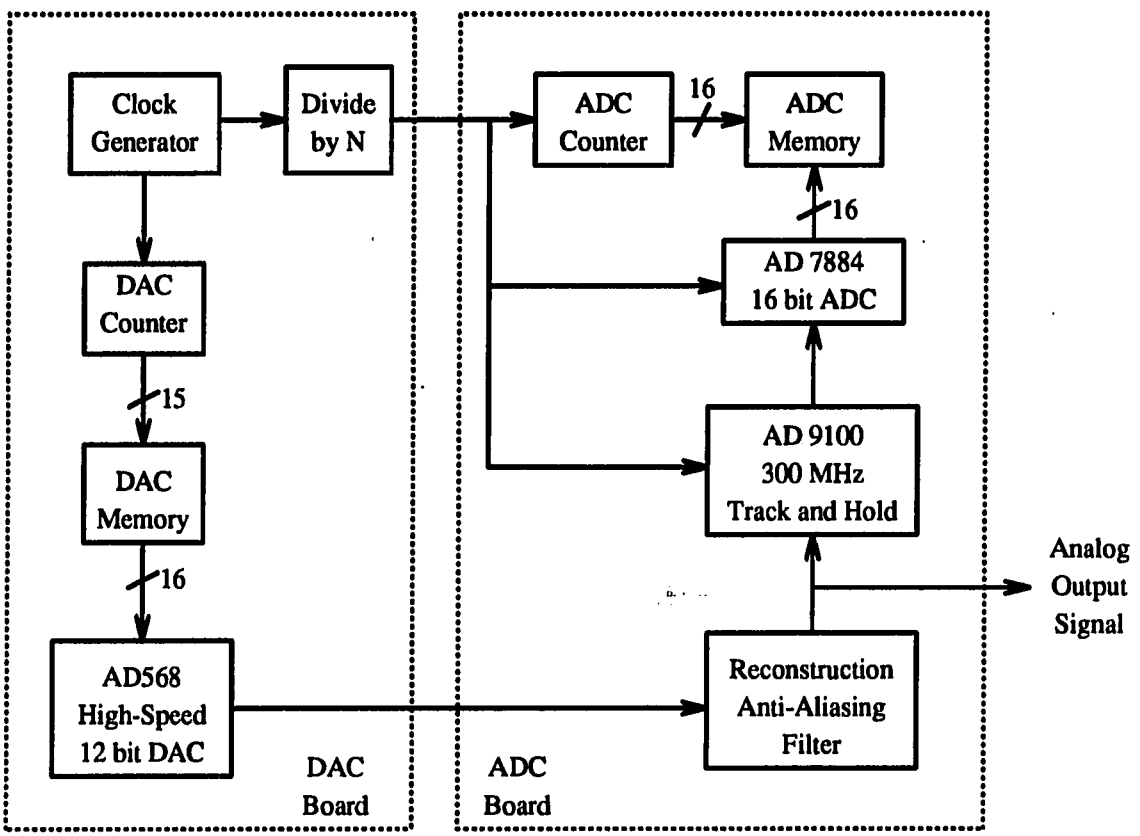

Figure 4.2: Simplified schematic of the ADC and DAC boards

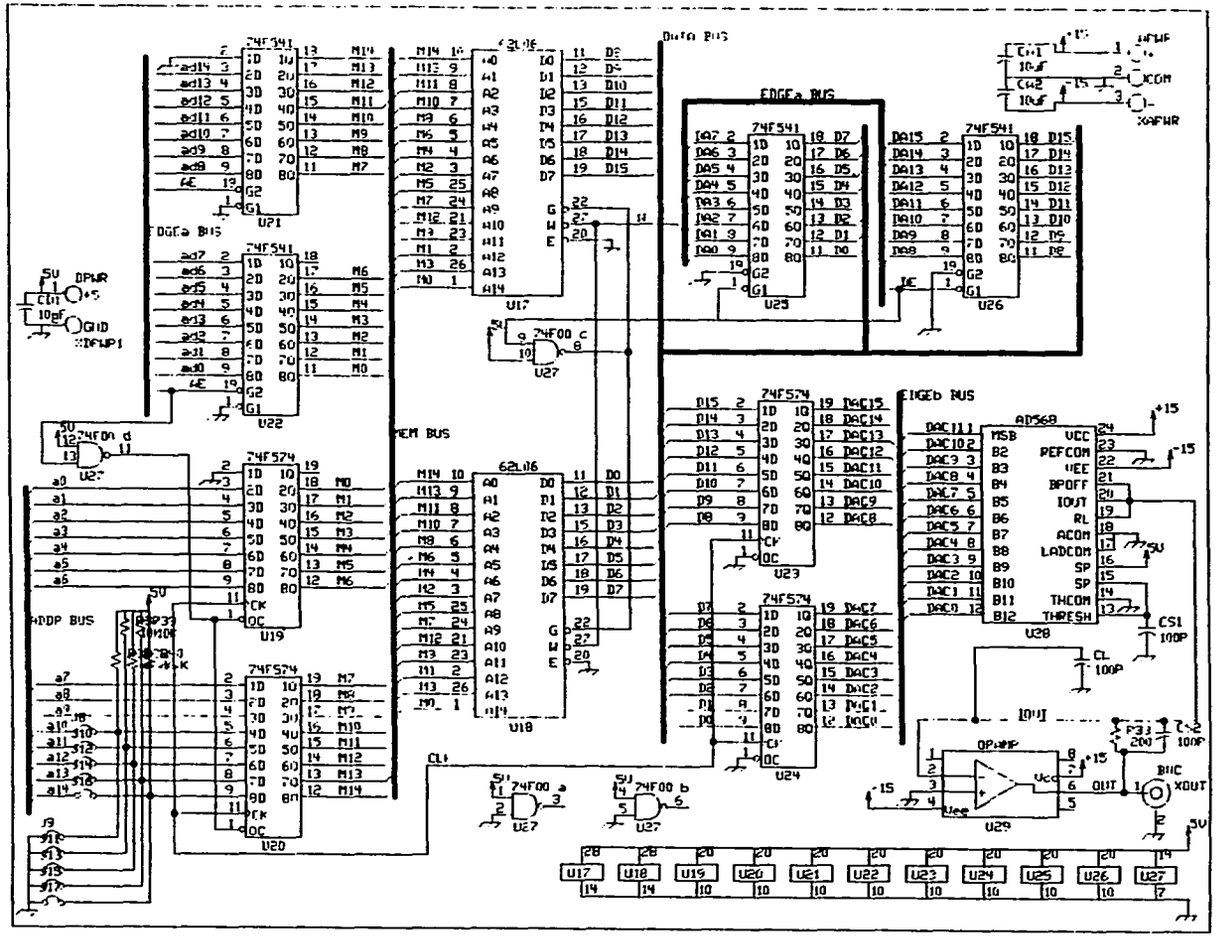

Figure 4.3: Detailed schematic of the generation circuitry showing the DAC, memory, and computer-controlled I/O circuitry

resistors). For this design the AD568 was configured to provide either an unbuffered, bipolar output of ±1.024 volts, or a buffered output of variable range (using the AD845 High-Speed Op Amp).

4.2.2 DAC memory

Two Motorola MCM62L06C 32K x 8-Bit Fast Static RAMs (Random Access Memories) were used for the storage and high-speed retrieval of the digital DAC input signal. They were chosen for their speed (20 ns address access time) and 8-bit word width (which keeps the chip count down).

A First-In, First-Out (FIFO) memory structure could have been chosen for its simplicity and low system part count, but this option has the drawback of a fixed record length, where the RAM length is controlled by the roll-over point of the counter. The RAM can also be partitioned by writing different signals to different sections of memory, and then only allowing the address counter to count through specific addresses.

The MCM62L06C is very simple to operate, with 15 address pins, 8 data I/O pins, 2 power and 3 control pins. The controls are: write/read ($\bar{W}$), output enable ($\bar{G}$), and chip select ($\bar{E}$). With 15 address lines, record lengths of 32,768 samples can be generated.

4.2.3 DAC address counter

To provide addresses at rates up to 50MSPS, four 74F169 binary counters with synchronous loading were used. The arbitrary position roll-over feature is realized by the circuit shown in Figure 4.4, and any length up to 65,536 can be generated (but only half can be utilized by the existing memory).

4.2.4 DAC data register

To control the data skew due to varying address access times (from bit-to-bit and sample-to-sample) a register is used to align and present the data from the memory to the DAC at exactly the sample rate. This function is provided by two 74F574

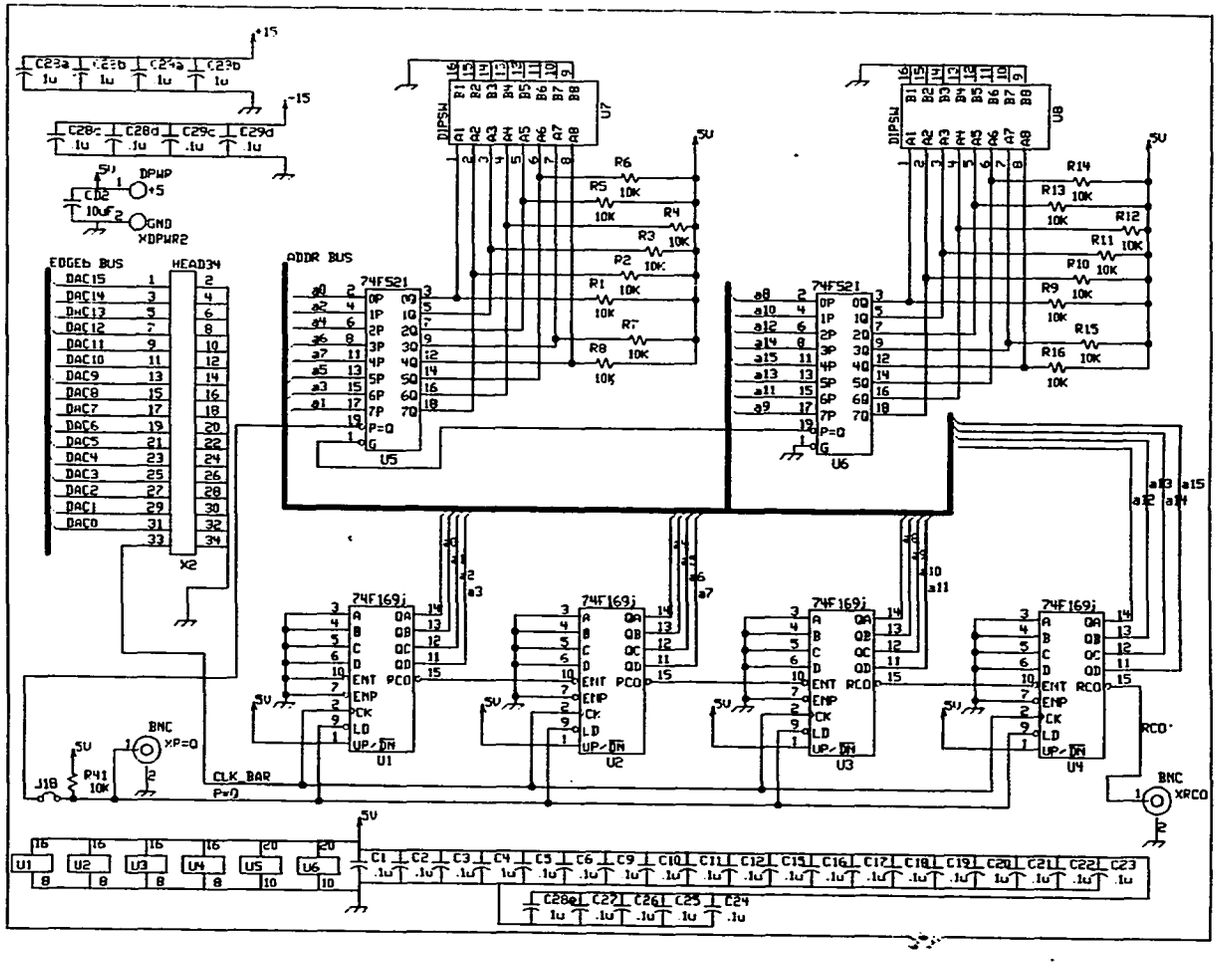

Figure 4.4: Schematic of the address counter and roll-over circuitry

Octal D Flip-Flops. The data skew can then be varied at the output of the '574s by appling different capacitive loads.

4.2.5 Bus control

To avoid contention for the address bus between the counter and the 68HC11, two sets of tri-state buffers (74F541) were necessary. Another set of buffers was used to free the data bus when the memory is driving the DAC (and not being loaded by the workstation).

4.3 Acquisition Circuitry

A detailed schematic of the analog potions of the board appears in Figure 4.5. All of the circuitry in this section runs at a much lower clock rate than the DAC circuitry (typically at $f_S/[L+1]$) by employing subsampling, so only the track-and-hold amplifier (THA) need handle the bandwidth of the DAC signals. A track-and-hold is simply a special case of sample-and-hold where the output is very well defined when not in hold mode.

4.3.1 On-board anti-imaging/anti-aliasing filter

The discrete RLC $5^{th}/7^{th}$-order elliptical filter shown as part of Figure 4.5 is provided on the ADC board to perform both the anti-image (or reconstruction) filtering for the DAC output signal, and the anti-alias filtering for the ADC input signal. The DAC system output is therefore actually on the ADC board. The correct circuit values for the filter are not those in the figure but were tailored for their performance below 20MHz.

This filter idealy provides 80dB of rejection beyond 10MHz, and has a pass-band of 5MHz (with a designed ripple specification of 3dB). The ideal and measured transfer functions of this filter are shown in Figure 4.6. There is a systematic attenuation of two due to this filter. To add flexibility, the filter can be bypassed in favor of an off-board, commercial coaxial filter.

Simulation predicted and testing verified that this filter is very sensitive to parasitics. Series resistance in the inductors, ground line inductance, and parallel board

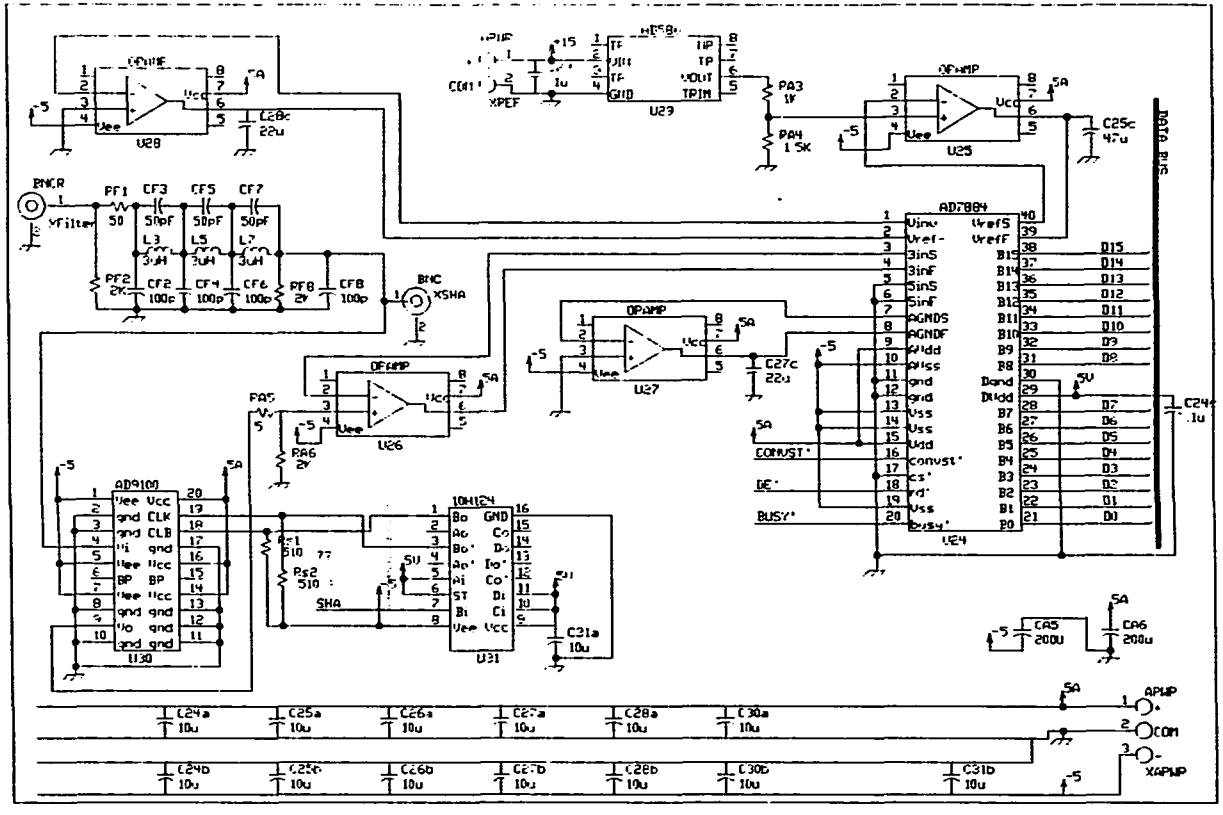

Figure 4.5: Detailed schematic of the analog portions of the acquisition circuitry

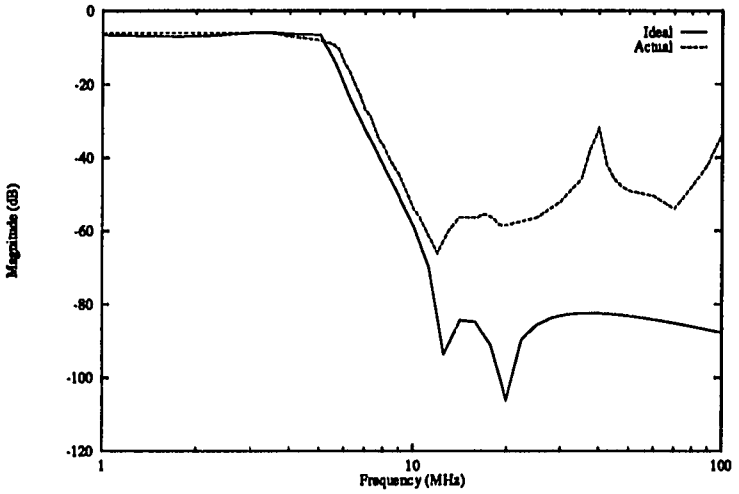

Figure 4.6: Ideal and measured filter transfer functions

trace capacitance seriously distort the pass-band and limit the stop-band rejection to 55dB between 10 and 30 MHz. Fortunately the algorithm is able to compensate for the the passband, as well as the lack of proper termination for the cable carrying the signal between boards (100 Ohm, not 50 Ohm, termination provides the best swing for the DAC). The spike in the frequency response seen at 45 MHz is most likely due to filter component nonidealities and electro-magnetic pickup.

To achieve improved immunity to electro-magnetic pickup, and to lessen cross-talk between filter nodes, the filter was laid out so that it could be surrounded by a grounded copper shield. Due to time constraints this was not implemented.

4.3.2 Track-and-hold amplifier

The highest performance component in the test system is the AD9100 Monolithic Track-and-Hold, which has a bandwidth of 250 MHz, and maintains a 80 dB spurious-free dynamic range when sampling signals of up to 10 MHz.

The AD9100 has a ±2 volt input range, larger than the DAC output signal (especially after attenuation due to the filter), so that close to 12 dB of dynamic range is sacrificed if the DAC board's output buffer is not used. This brings the

signal closer to the noise floor, but should not significantly change the SFDR.

The clock signal for the AD9100 must be complementary emitter-coupled logic (ECL) and is provided by a MC10H124 TTL-to-ECL translator.

4.3.3 Analog-to-digital converter

The AD7884 16-bit, High-Speed Sampling ADC resamples the held signal from the THA and converts it into a digital value in 5.3 us. The reason for the resampling is that the droop rate of the AD9100 is too high to allow reliable 16-bit accuracy for much more than about 300 ns.

The AD7884 has a significant amount of external circuitry (as shown in Figure 4.5), including a voltage reference (Analog Devices AD586) and four operational amplifiers. The amplifiers are needed for inverting the reference (Analog Devices AD711) and to ensure accuracy of the input, ground and reference pin potentials (Analog Devices OP-07 and AD845).

The AD7884 has a SFDR of well over 85 dB when driven by an accurate THA, because the major contributor to its distortion is its own SHA, which in this case is sampling a DC signal. The input range is ± 3 volts, so again some dynamic range is sacrificed.

4.3.4 ADC address counter and memory

The counter and memory on this board function in the same manner as on the DAC board, but implement a write to memory and are clocked at a much lower speed. The same devices are used for simplicity, but dip switches were replaced by five jumpers. The roll-over of the counter signifies that the memory is full and halts the write operation. Figures 4.7 and 4.8 are schematic diagrams of the address counter, memory, and bus control circuitry.

4.3.5 Bus control

The address bus of the ADC board is controlled in the same way as on the DAC board. The data bus is somewhat simpler in this case because, to avoid contention

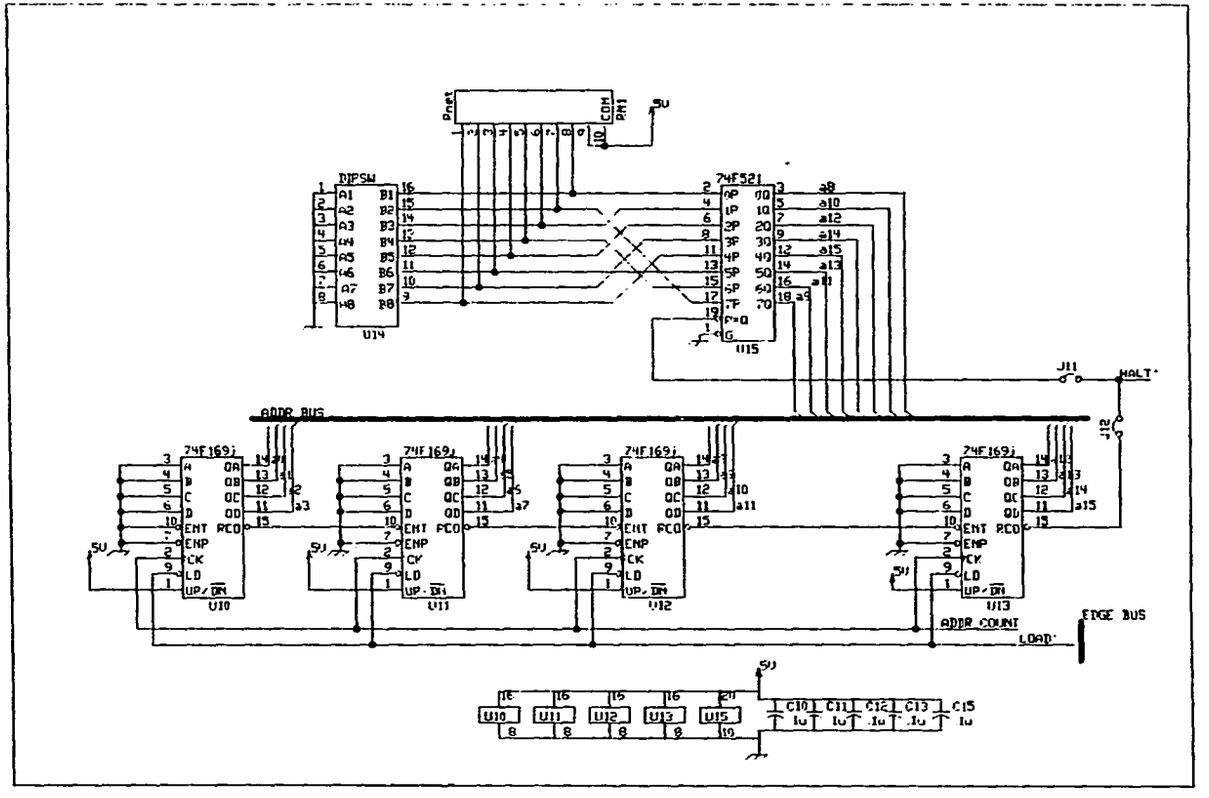

Figure 4.7: ADC board address counter and "memory full" detection circuitry

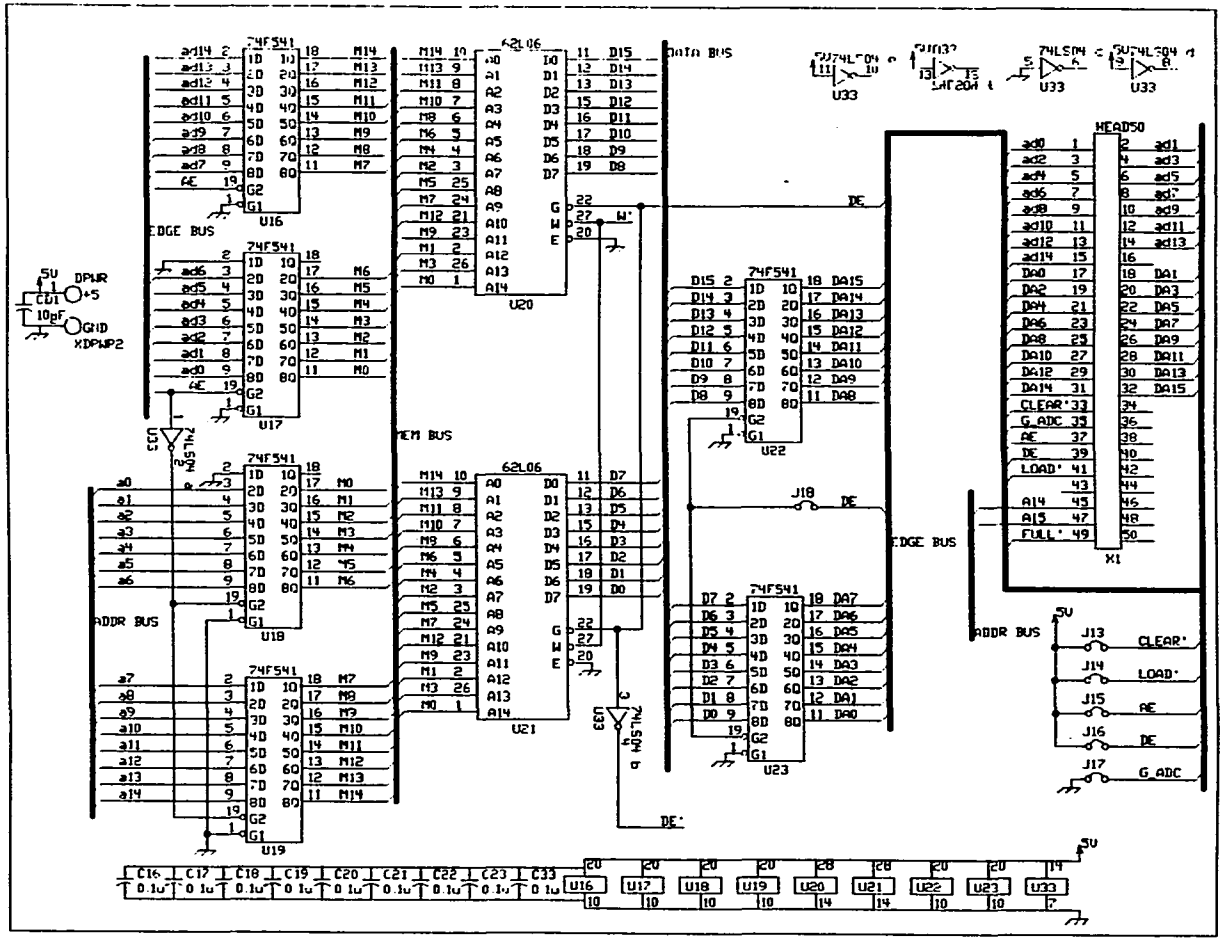

Figure 4.8: ADC board memory and bus control

when reading from ADC memory, the AD7884 has its outputs tri-stated, or in high-impedance mode. Because of this, only one set of buffers is necessary, those that drive the data cable to the 68HC11.

4.4 Clock-Generation Circuitry

The task of the clock generation circuitry is to generate a high-frequency square wave ($\approx$ 50 MHz) from a low-jitter crystal to drive the DAC, and then divide that signal down in order to provide a sampling clock for the ADC board (Figures 4.9 and 4.10).

Typically the ratio between these two clocks will be one more than the DAC-memory record length, or $L + 1$, to provide a convenient beat frequency. To add system flexibility, jumpers are used so that independent external sources can be utilized.

The basic component is the 74F169 cascadable, 4-bit up/down counter. This part has a chip enable (to halt clocking of one or both boards) and a selectable divide ratio. This ratio is set by dip switches and is controllable from 1 to 65,536 for the ADC (4 dip chips). The crystal used was a 40 MHz CMOS/crystal hybrid chip, with a jitter specification of under 50 psec.

The clocking circuitry is split between the two boards. Each board also needs some additional logic to provide the clocks at the right times (e.g., the ADC clock should be applied to the sampler for the first time when the DAC counter rolls over so that there is no perceived phase error between the two digital signals in memory). This control operation is known as *gating* the clocks, and is performed by circuitry shown in Figures 4.9 and 4.10.

There is a concern when dividing down a clock signal that the jitter of the edge will be increased, leading to a timing error in the sampling. For this reason the very fastest logic is used, even for the ADC board clock circuitry which runs relatively slowly. The total logic seen by the clocking signal prior to being applied to the THA is four gates and 2 flip-flops, which is relatively small considering the frequency division operation being performed.

The clocking and control circuitry of Figure 4.9 is needed to provide the proper

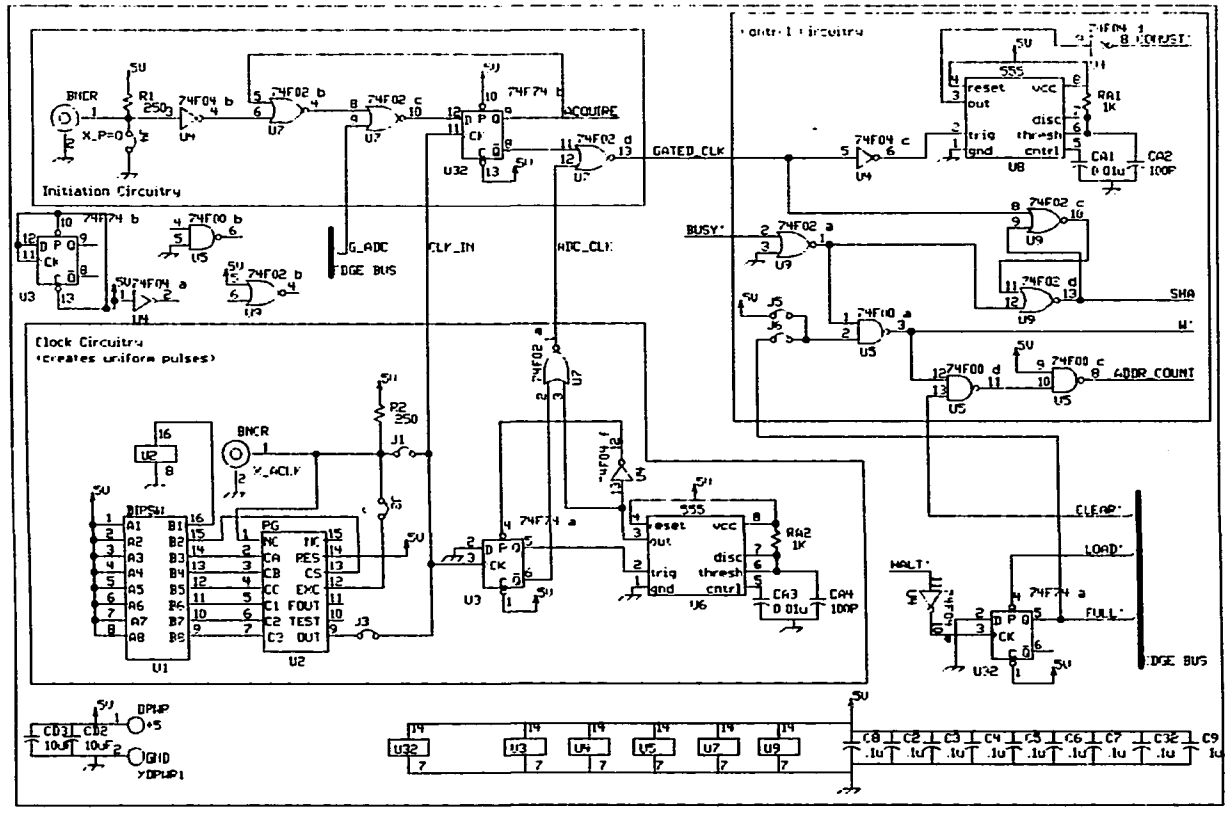

Figure 4.9: Schematic of the clock and control circuitry resident on the ADC board

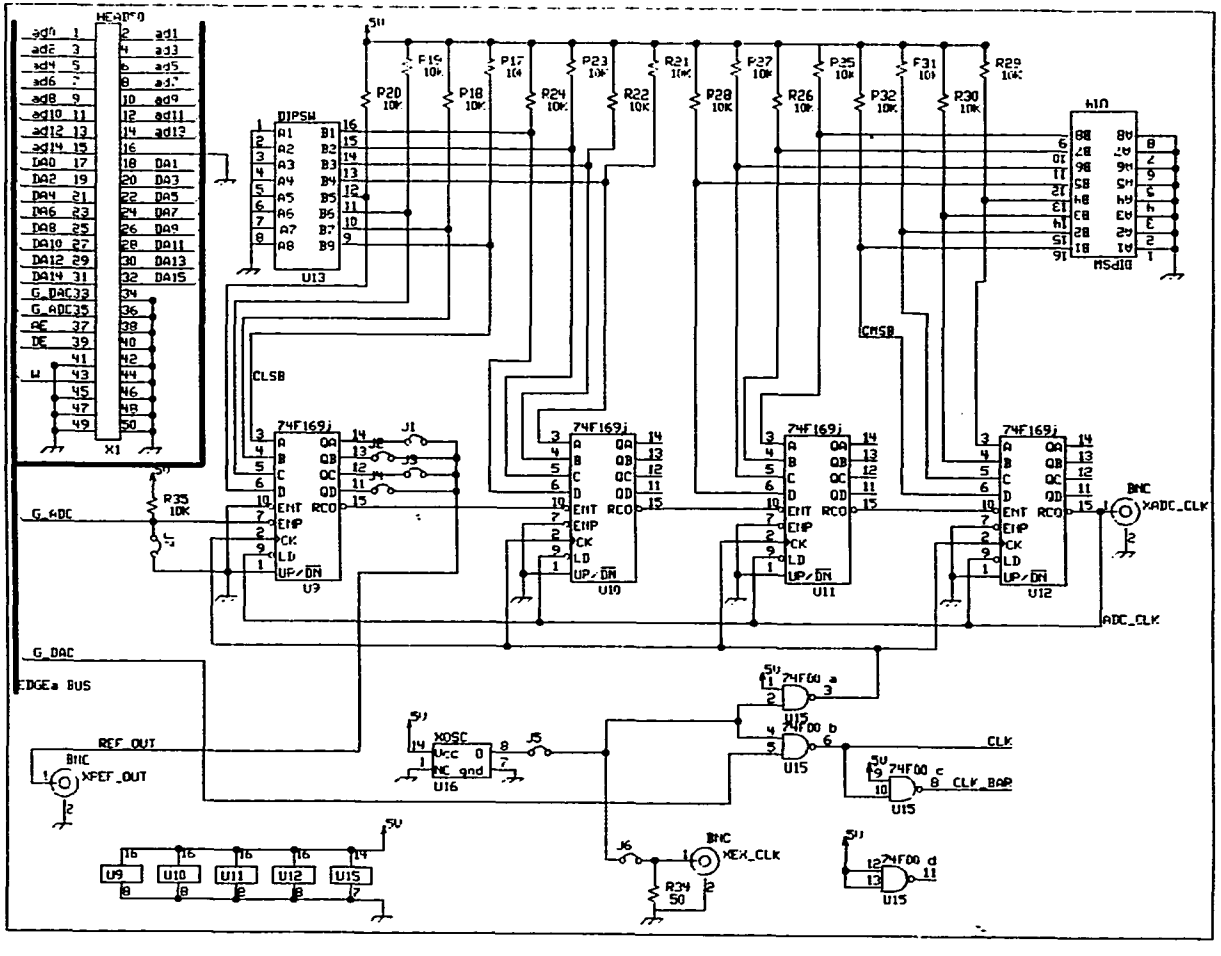

Figure 4.10: Schematic of the clock generation circuitry resident on the DAC board which divides the high-frequency clock to provide the ADC clock

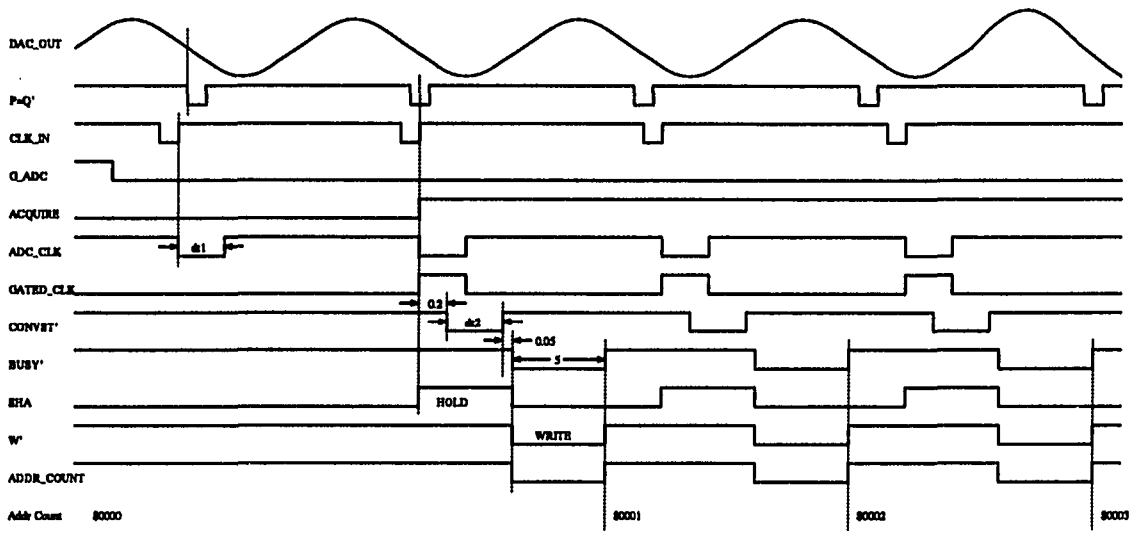

Figure 4.11: Timing diagram of the control circuitry on the ADC board showing acquire initiation (all times in micro-seconds)

timing for the hold functions of the AD9100 THA and the AD7884 ADC, as well as the write function and the clocking of the address counter. Timing diagrams for this circuitry appear in Figures 4.11 and 4.12.

4.5 Evaluation Instrumentation

To verify that the algorithm was indeed correcting for DAC and filter nonidealities, the HP8569A and Textronics TEX2710 spectrum analyzers were used to determine the purity of the signal in the frequency domain. They were connected as closely as possible to the input of the THA to ensure that they and the ADC components saw the same signal (including cable attenuation and ADC board effects).

For additional evaluation and debugging purposes, the Textronics 7504 300 MHz oscilloscope was used to monitor the signals in the time domain.

4.6 Computer Control and Algorithm

As mentioned earlier, the FFT processing, operating mode, and predistortion algorithm are all controlled by the HP workstation. The interface to the MC68HC11

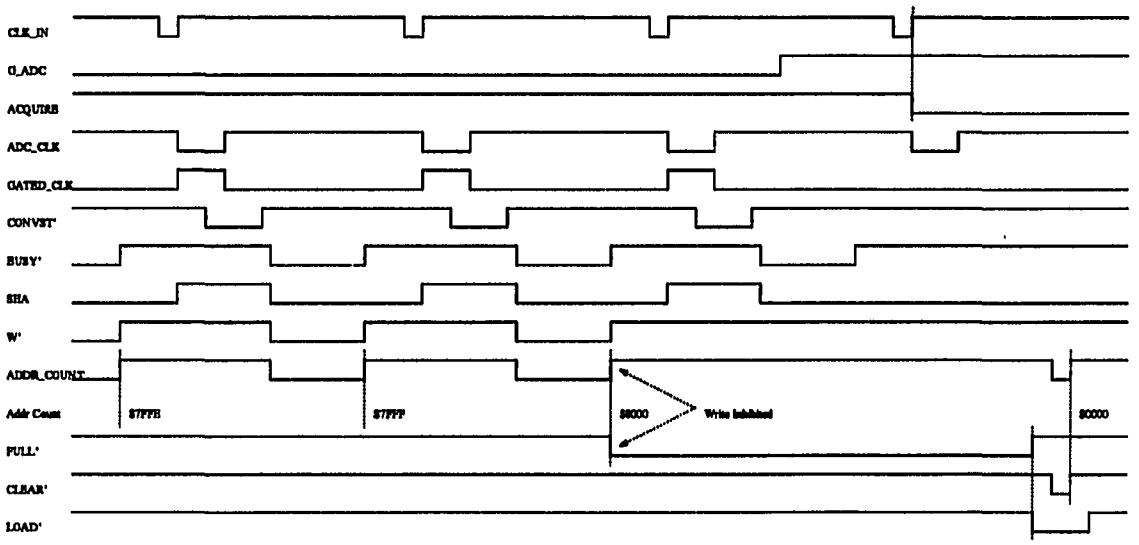

Figure 4.12: Timing diagram of the control circuitry on the ADC board showing the functions of "FULL" and "WRITE" inhibits

board is via the RS232 asynchronous serial port. The MC68HC11 is connected to both the ADC and DAC boards by 60 lead ribbon cable.

The algorithm housed in the workstation can be represented by the block diagram shown in Figure 4.13, which is simply a variation of that discussed in Chapter 3.

4.7 Results

This section is a survey of the spectral improvement achieved by the predistortion algorithm using the test system just described. Unless otherwise noted in the text, the following defaults apply. The DAC update rate is set to $f_S = 20$MSPS, and the DAC board memory length is 256 samples. The DAC has 12 bit resolution and a buffered output swing of ± 1.2V. The DAC clock is divided by 257 to yield a 12.85u second ADC clock period, or a sample rate of $f_{ADC} \approx 77.8$KSPS. The algorithm defaults to capturing 10 copies of the signal, each 256 points in length. The FFT is performed on a point-by-point average of this signal, yielding a 256 bin output. The bandwidth of interest then limits the correction to the first 50 bins (4MHz). These

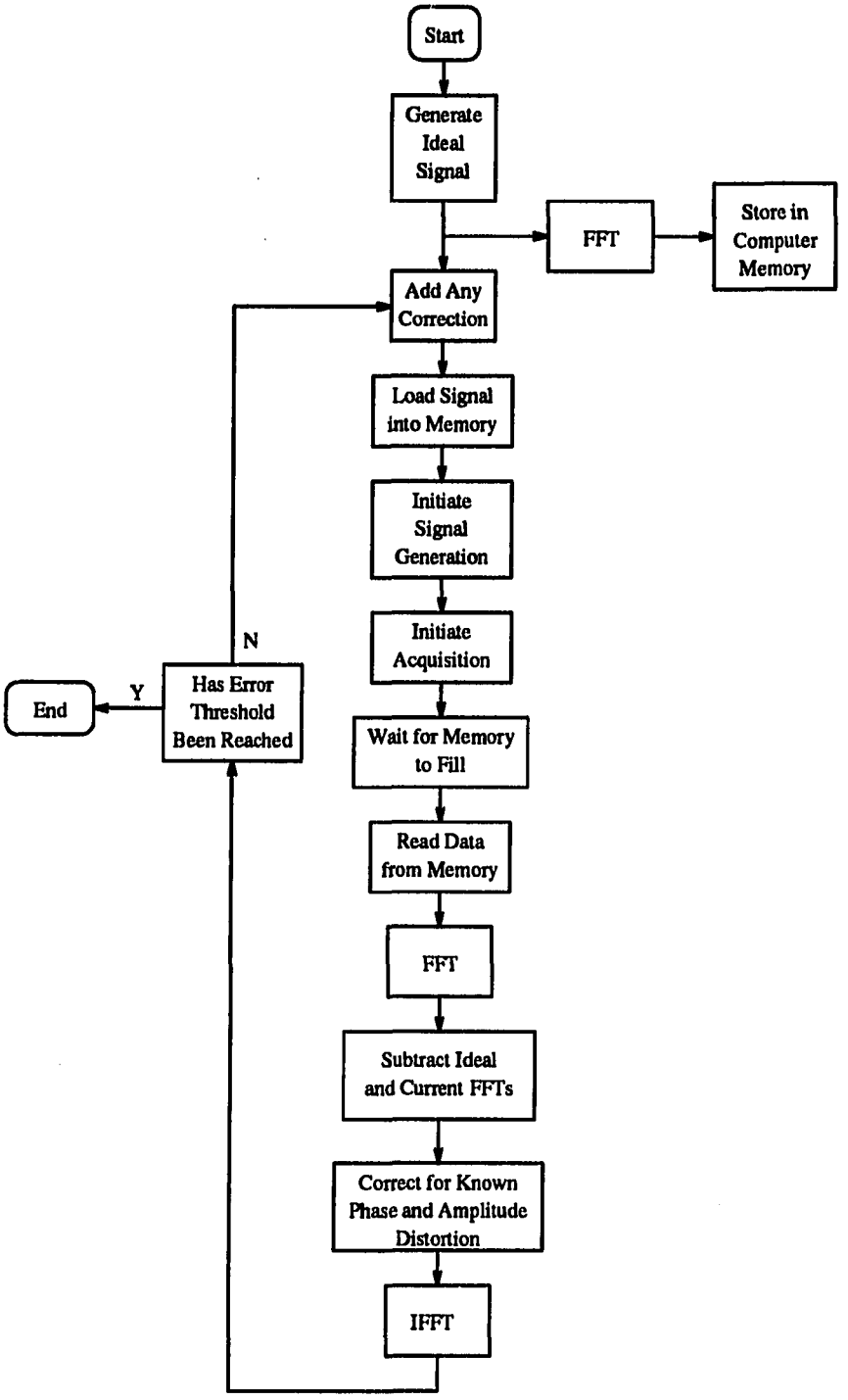

Figure 4.13: Block diagram of the C-based test program

Table 4.1: System and algorithm defaults

Section	Setting	Default Value	Comments
DAC	Update Rate (f_s)	20 MHz	50 nsec period
	Memory Length (L)	256	
	Resolution	12 bits	
	Output Mode	$\pm$1.2 V	Buffered
ADC	Resampling Period	12.85 usec	every 257 points
Algorithm	FFT Window Length	12.8 usec	256 points
	Number of Averages	10	
	Bandwidth of Interest	4 MHz	$f_s/5$ 50 bins

defaults are summarized in Table 4.1.

The algorithm was used in the laboratory to generate three different types of signals: constant DC voltages, single-tone sine waves of various frequencies and amplitudes, and multi-tone signals. A summary of the test results appears in Table 4.2.

4.7.1 Constant DC voltages

The first test consisted of loading all 256 DAC memory positions with the code corresponding to a 1.0V output and running both the DAC and ADC boards to acquire the spectrum. The resulting FFT contained only two components above -85 dB. The first was 0.077 dB in bin 0, indicating an offset of 8.9mV, or an error 41.0 dB below the signal level. The other component, in bin 1, was 69.3 dB below the signal. Because the DAC code remained constant, the latter must be a result of digital feedthrough from the counter circuitry.

After four iterations of the algorithm the DC offset was reduced to 37uV (-88.6 dB) and the SFDR increased to 77.8 dB (contained in bin 4). Although the worst spectral errors were reduced, in general the spectral content of all other bins increased, resulting in many bins containing components in the -80 to -85 dB range.

Table 4.2: Summary of predistortion test results

Signal		SFDR (dBc)			
Frequency (KHz)	Amplitude	Initial	Final	Iter.	Comments
0.0 (DC)	1.0 V	69.3	77.8	4	offset removed
78.125	1.0 V	74.4	80.4	9	bw is 30 bins
156.25	1.0 V	66.3	76.4	5	
234.375	1.0 V	64.0	80.0	5	bw is 30 bins
781.25	1.0 V	52.5	71.5	8	
1250.0	1.0 V	46.4	70.6	9	
1250.0	0.1 V (-20dB)	37.2	52.3	7	
234.375/1250	0.2 / 0.4 V	57.4	71.3	8	two-tone
78.125 Sawtooth	1.0 V	–	–	3	see plot

4.7.2 Sine waves

In this section the DAC memory is initially loaded with an ideal digital sine wave, meaning that it has been quantized by rounding to the nearest 12 bit code, but not precompensated or predistorted in any way. The iteration numbers which appear in Table 4.2 refer to the cycle which achieved the highest SFDR, and ignore DC offset and gain errors.

Table 4.2 contains the predistortion results of 1V (2V ptp) sine waves at five frequencies between 78.125 KHz ($f_s/256$) and 1.25 MHz ($16f_s/256$). In each case SFDR numbers increased by more than 6 dB, yielding values above 70 dB. In two cases (78.125 and 234.375 KHz) the bandwidth of interest was reduced from 50 to 30 bins, enabling the algorithm to produce SFDRs over 80 dB. Spectrums before and after predistortion for the signals at 234.375 KHz, 781.25 KHz, and 1.25 MHz appear in Figures 4.14, 4.15, and 4.16, respectively. The drastic deterioration in the initial SFDR with signal frequencies indicates that the dominant high-frequency DAC errors are dynamic in nature.

It is interesting to note the significant differences in the initial spectrums of Figures 4.15 and 4.16. The 1.25 MHz sine has much higher harmonic distortion, but the bins in between these harmonics have much lower amplitude components

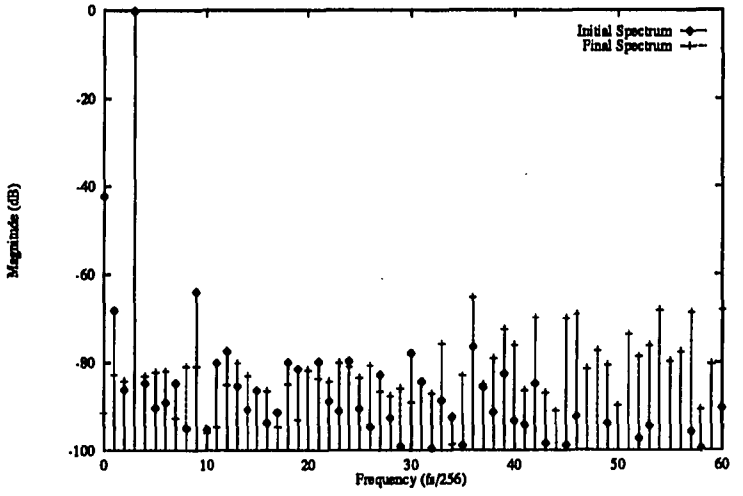

Figure 4.14: Spectrum of 234.375 KHz ($3f_s/256$) sine wave before and after predistortion

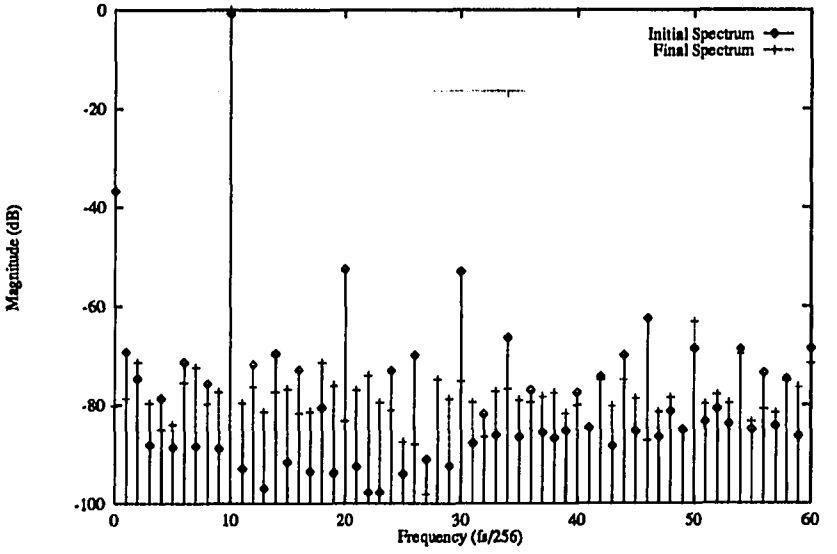

Figure 4.15: Spectrum of 781.25 KHz ($10f_s/256$) sine wave before and after predistortion

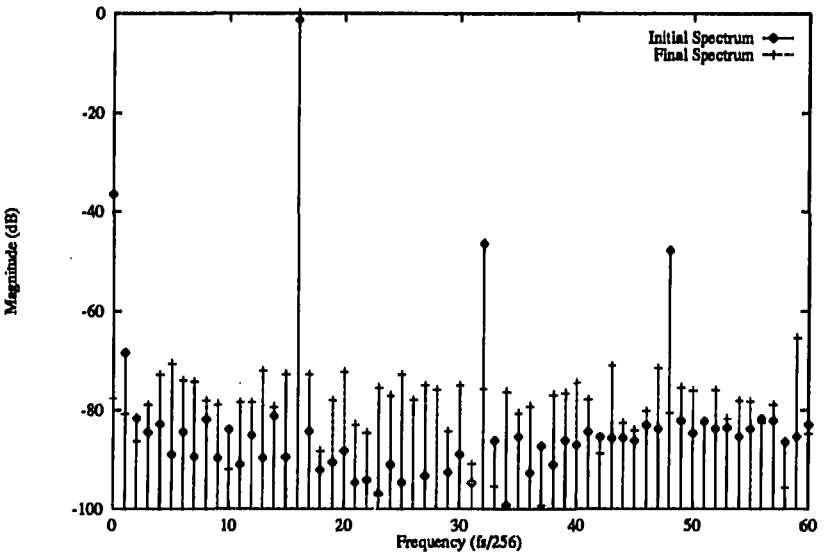

Figure 4.16: Spectrum of 1.25 MHz ($16f_s/256$) sine wave before and after predistortion

than the 781.25 KHz spectrum. The 781 KHz signal's distortion seems to be spread more evenly over the bins. This is understandable when you realize that for the 1.25 MHz case the digital repetition period is 16, meaning that the exact same codes are repeated every 16 clock cycles. If the analog signal is truly periodic on 16T, then there can be no spectral components between integer multiples of 16 in the 256 point FFT. The digital repetition rate of the 781 KHz signal is 128, leading to components in every other bin.

Figure 4.17 shows the DAC's SFDR, as measured at the completion of each iteration, for various input frequencies. A similar plot of DC offset appears in Figure 4.18. Note how the values tend to vary with iteration. Since the signals from all past iterations can easily be stored in memory, and therefore re-used at any time, the algorithm's true performance is actually the best signal of all previous iterations. This means that the algorithm's performance is a monotonically increasing function of iteration number. If this aggregate performance idea is applied to the data of Figure 4.17, Figure 4.19 results. The algorithms performance on worst case error versus iteration number appears in Figure 4.20. This last figure includes gain and

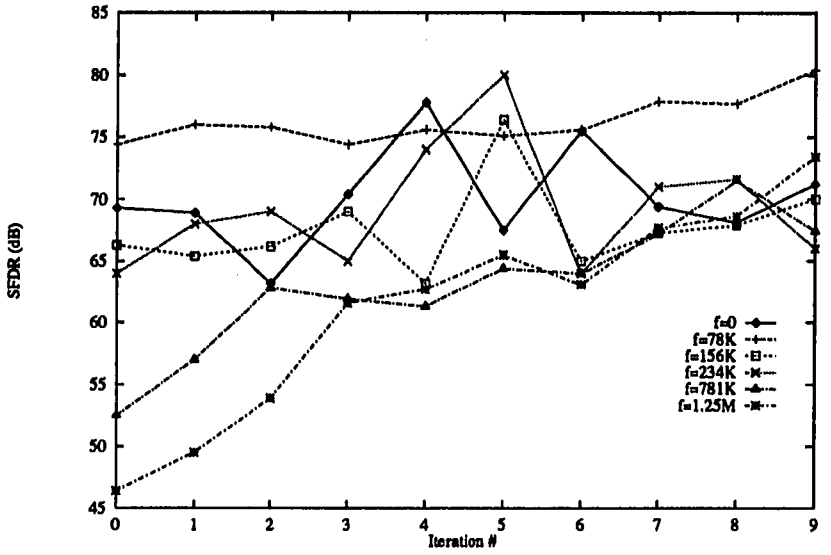

Figure 4.17: SFDR versus iteration number for various sinusoids

offset errors with SFDR when determining the worst error.

When the amplitude of the signal at 1.25 MHz was dropped by 20 dB, the algorithm lowered the dominant error spur by about 15 dB to about 72.3 dB, yielding a new SFDR of 52.3 dBc (with respect to the new lower signal).

4.7.3 Multi-tone signals

The multi-tone signals tested also appear in Table 4.2. The first is a mix of 1.25 MHz and 234.375 KHz sines at 0.2 and 0.4 Volt amplitudes, respectively. In the initial spectrum, intermodulation products can be detected around the 1.25 MHz sine's second harmonic, as shown in Figure 4.21.

Figures 4.22 and 4.23 are the frequency and time domain plots of a 0.5 V (1V ptp) sawtooth signal with a period of 12.8u seconds. Inside the 50 bin bandwidth of interest, the algorithm was able to remove virtually all of the errors except for a one-bin spike at close to 3 MHz (its amplitude is about -40 dB). Although the cause of this error was never verified, it is believed to be the result of an error in the phase correction mapping, leading to an instability at this frequency. Multiple acquisitions

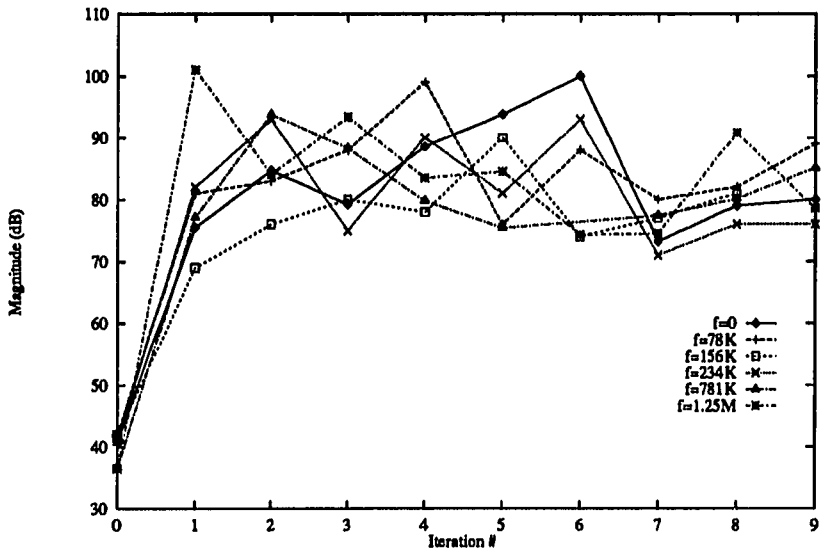

Figure 4.18: DC error versus iteration number for various sinusoids

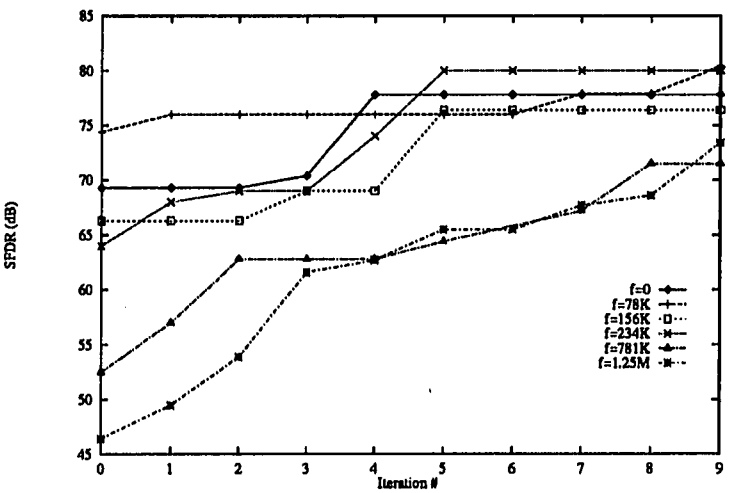

Figure 4.19: Algorithm's SFDR performance versus iteration number

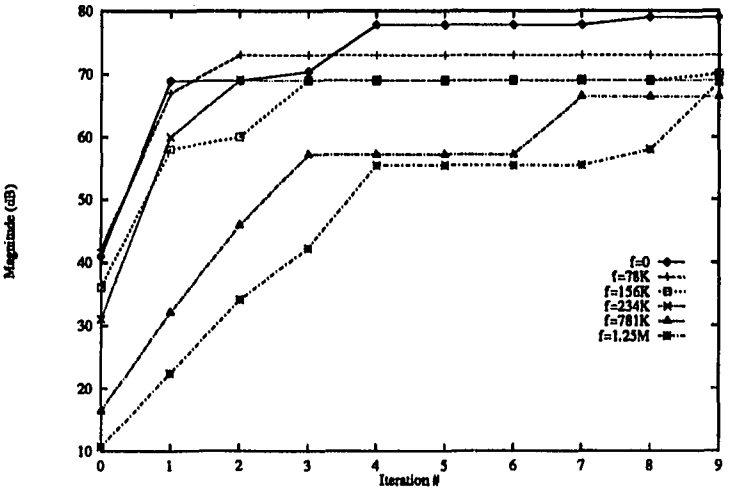

Figure 4.20: Worst case error performance versus iteration number (includes gain and offset)

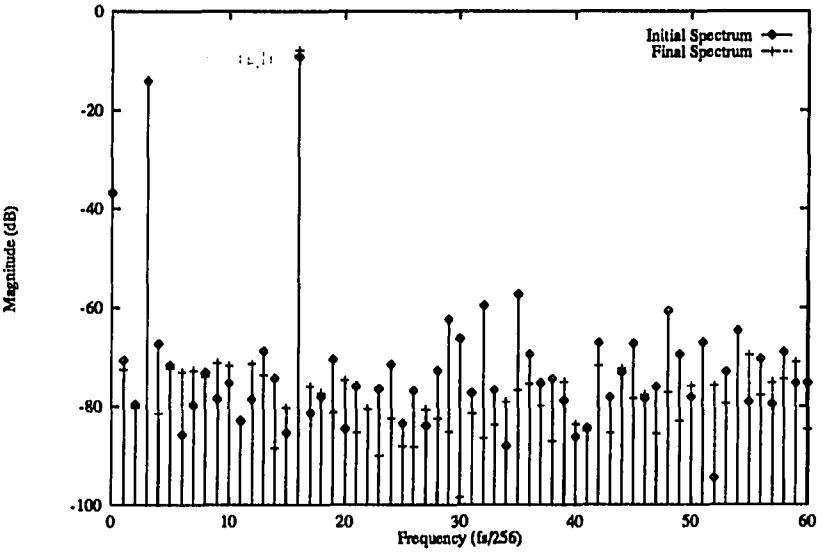

Figure 4.21: Spectrum of a two-tone signal before and after predistortion

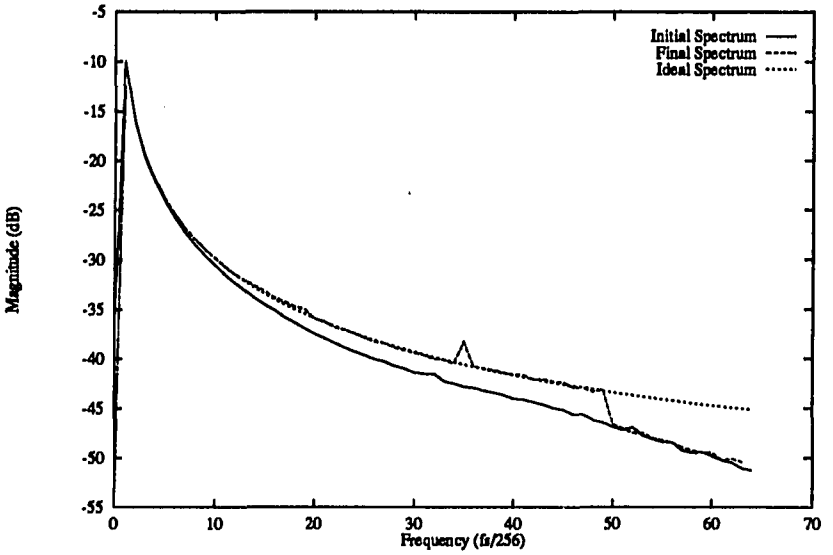

Figure 4.22: Spectrum of ramp before and after predistortion (note instability)

of the same digital code indicated that the signal was not external interference, and its lack of appearance in other spectrums ruled out a clock related source.

4.7.4 System performance

In all laboratory tests the predistortion algorithm improved the SFDR by 6 dB or more, and in the cases where the bandwidth of interest was reduced to 30 bins the SFDR values topped 80 dB. These results indicate that the algorithm can extend the accuracy of high-speed DACs by as much as two bits without impacting update rate. The only penalty for these improvements is the limitation in bandwidth of interest, which is a penalty most DDS systems already impose for other reasons (filter transition band, output circuitry bandwidth limitations, etc.).

The increase in harmonic distortion with input frequency indicates that the dominant errors in the test system were dynamic. The algorithm appeared to handle these errors as well, but the final SFDR and iteration numbers for high-frequency signals indicate a limit to this implementation of predistortion. The DC test indicates that the algorithm is effective against digital feedthrough, an accomplishment previously

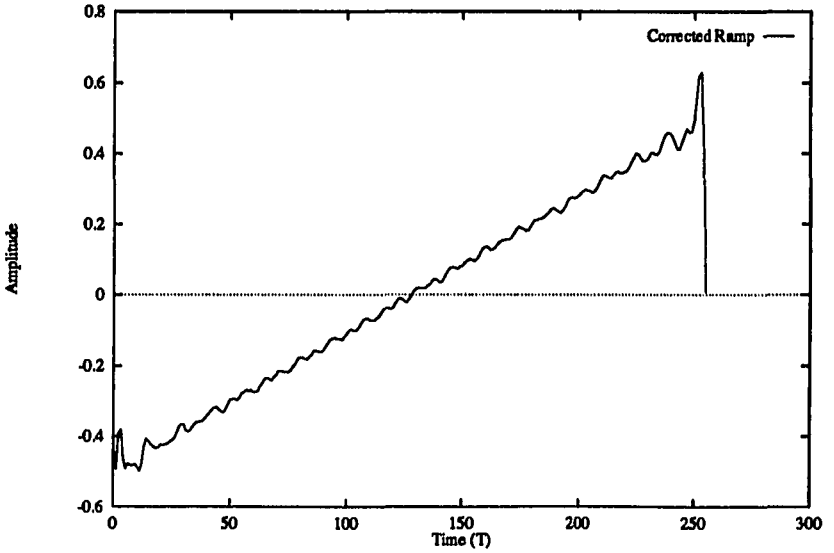

Figure 4.23: Time-domain corrected ramp signal

unreported.

The speed of the algorithm is basically determined by the number of iterations multiplied by the FFT processing time. This implementation of the algorithm was limited by the time necessary to up- and down-load the signal from the boards through the serial interface, but other test configurations could easily reduce this time with a full parallel interface. The total signal acquisition time of the ADC board is approximately the THA/ADC acquisition time multiplied by the total number of points to be sampled. This is an approximation because to assure signal coherence from iteration to iteration requires the system to wait for the ADC sampling edge to align with the start of the DAC memory.

To determine the acquisition time for a single iteration, multiply the following quantities: ADC subsampling ratio, DAC memory length, signal averages, and DAC clock period. For most of the simulations presented here (with 256 FFT points and 10 signal averages) the acquisition time was:

$257 \times 256 \times 10 \times 50\text{nsec} = 32.9\text{msec}$.

The limitations to this test configuration are:

- Algorithmic — This algorithm, which tries to correct all of the spectral errors above a certain threshold, is not the most efficient one. This can be seen in the 1.25 MHz spectrums where most of the spectral components increase during the first few iterations. An algorithm which is weighted more heavily towards the worst few errors would probably be more efficient.

- Line Interference — A small amount of 60 cycle interference could be detected at the output of the DAC, even when not being clocked. To reduce its spectral impact the signal was averaged over a period of approximately 1/60 of a second.

- Number of DAC Bits — Simulation indicated that additional DAC bits, which simply provided more possible output levels, increased the resolution of the system even if the accuracy of the DAC was not increased.

- Limited Acquisition Resolution — The algorithm managed to correct most signals to close to the ADC boards resolution (about 80 dB).

- Sampler Power Modulation — It was discovered that the ADC's resolution was reduced slightly by the operation of the THA (by about 6-8 dB). It is believed that this was due to the large power supply requirements of the THA affecting the ADC's signal fidelity.

These errors where detected in part by comparing the FFT spectrums to those provided by the stand alone spectrum analyzers.

5. CONCLUSIONS

This research has shown that digital predistortion can greatly increase the dynamic range of high-speed DACs. In addition to drastically reducing most static errors, predistortion also reduces the effects of dynamic nonlinearities, which are the dominant error source in DAC-based signal-generation systems.

Communications systems are continuously demanding higher spectral quality and greater flexibility from signal generation circuitry, and the instrumentation necessary to test such equipment requires even greater performance. As this trend continues, direct digital synthesis will dominate because the signal can originate in, and be controlled by digital circuitry with its inherent wide dynamic range. Digital control simplifies arbitrary signal generation, and allows faster, cleaner phase and frequency shifts than analog circuits.

As the major contributor to DDS system distortion, high-speed DACs command a great deal of attention in waveform synthesis. Research should concentrate on modeling and curing transition nonlinearities, with special emphasis on data skew and digital feed-through. As the mechanisms for transition distortion become better understood, methods for removing them will be discovered.

Chapter 1 and 2 show how large the effects of data skew can be and provide a crude method for estimating spectral errors based on time-domain information. This work should be expanded to include charge injection and digital feed-through. The ability of output samplers and return-to-zero techniques can then be evaluated based on these findings and the analysis at the end of Chapter 2. The point of diminishing returns of segmentation and thermometer decoding can be found in the same way.

Although methods for easing transition distortion exist, they currently come at a high price. They all sacrifice speed, power, and area for wider dynamic range. At some point DAC complexity becomes detrimental to the signal, indicating that a

new approach should be employed. Both integrated samplers and system-level signal conditioners should be investigated.

In fact, this work shows that output sampling and signal preconditioning may work best together. Sampler distortion is shown to be more predictable than that of a distributed converter system. If a model could be developed for these errors, then predistortion could be implemented without a feedback network, allowing real-time correction and eliminating the major drawback to the current predistortion algorithm. This technique has already been applied successfully to the limited case of sinx/x distortion. Such a system would have all of its complexity in a nonlinear digital prefilter, greatly reducing the demands on the analog circuitry. The sampler settling should slow the maximum update rate of the converter by less than a factor of two.

The predistortion algorithm demonstrates that most forms of distortion can be eliminated from the output signal if the proper input is applied. Although difficult and time consuming in the face of unpredictable and poorly-behaved error sources, a feedback loop is effective at finding the proper signal, as the results of simulation and laboratory testing demonstrate. Unfortunately this delay makes the algorithm inappropriated for real-time systems, but applications do exist, chiefly in production and lab testing environments. In these areas feedback may be more attractive because the output spectrum can be continuously monitored, and the evaluation circuitry is readily available and not considered overhead.

The laboratory tests in Chapter 4 show how effective predistortion can be. The dominant error spur of a state-of-the-art commercial DAC, whose distortion was dominated by dynamic errors, was pushed down by as much as 27dB. Improvements of 6dB were found in virtually all cases. When the bandwidth of interest was limited to approximately one tenth the update rate, 80 dB SFDR numbers were recorded. The greatest improvements were seen where the initial spectral errors were at their greatest, indicating that the limitations to this implementation were the number of DAC bits and the accuracy of the acquisition circuitry.

This data suggests that state of the art DACs, intended to be run at update rates allowing settling to 1/2 LSB, and with AC and DC specifications currently available, should achieve SFDR performance equivalent to an ideal DAC with 1 to 2

bits of greater resolution. Stated another way, an N bit DAC with state of the art performance should achieve N+1.5 bits of ideal performance. This assumes access to the required acquisition circuitry and the addition of extra bits in the DAC (but the accuracy performance remains unchanged).

These results do not come at the price of update rate, but only at the sacrifice of signal bandwidth which is typically not used by DDS systems any way. Chapter 4 also demonstrated that dynamic errors were at least partially corrected, as was digital feedthrough, something previously unreported.

The results of this work also provide important insight into the origin of many distortion components, as well as the advantages of trading bandwidth for dynamic range as is routinely done in lower-frequency oversampled converters.

It is not yet clear what the true resolution limitations to the algorithm are, but simulations from Chapter 3 indicate that they may reside in unpredicted DAC discontinuities, as demonstrated by positive DNL simulations.

REFERENCES

[1] J. Studders, "Phase accumulator for direct digital synthesis resolves to within 0.7Hz," *Analog Dialogue*, vol. 25, no. 1, pp. 10–11, 1991.

[2] M. Thompson, "Low-latency, high-speed numerically controlled oscillator using progressions of states technique," *JSSC*, vol. 27, pp. 113–117, January 1992.

[3] A. Blanchard, *Phase-Locked Loops - Application for Coherent Receiver Design.* New York, NY: John Wiley & Sons, 1976.

[4] T. L. Brooks, *Correlated tuning of high-frequency integrated continuous-time filters.* Master's Thesis: Texas A&M University, May 1992.

[5] R. L. Pickholtz, D. H. Schilling, and L. B. Milstein, "Theory of spread-spectrum communications - a tutorial," *IEEE Transactions on Communications*, vol. COM-30, pp. 855–884, May 1982.

[6] J. Studders, "12-bit DACs optimized for direct digital synthesis," *Analog Dialogue*, vol. 23, no. 4, p. 6, 1989.

[7] T. H. Higgins, "Analog output system design for a multifunction synthesizer," *Hewlett-Packard Journal*, pp. 66–69, February 1989.

[8] M. D. Talbot, "Digital waveform synthesis IC architecture," *Hewlett-Packard Journal*, pp. 57–62, February 1989.

[9] J. Gallant, "Devices refine the art of frequency synthesis," *EDN*, p. 95, November 9, 1989.

[10] S. Pizzi, "The NCMO: an RF modulation breakthrough," *BME*, pp. 80–83, December 1988.

[11] *Analog Devices Data Conversion Reference Manual*, vol. 1. Boston, MA: Analog Devices, INC., 1992.

[12] D. H. Sheingold, *Analog-Digital Converter Handbook.* Prentice-Hall, 1986.

[13] W. Kester, "Test video A/D converters under dynamic conditions," *EDN*, August 18, 1982.

[14] S. Wayne, "Getting the most from high resolution D/A converters," *Electronic Products*, December 12, 1983.

[15] W. M. Siebert, *Circuits, Signals, and Systems.* Cambridge, MA: The MIT Press, 1986.

[16] M. Schwartz, *Information Transmission, Modulation, and Noise.* New York, NY: McGraw Hill Book Company, 3 ed., 1980.

[17] B. C. Kuo, *Automatic Control Systems.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 4 ed., 1982.

[18] H. Nicholas and H. Samueli, "A 150-MHz direct digital frequency synthesizer in 1.25-uM CMOS with -90dBc spurious performance," in *ISSCC Digest of Technical Papers*, vol. 34, pp. 42–43, 1991.

[19] R. A. Roberts and C. T. Mullis, *Digital Signal Processing.* Reading, MA: Addison-Wesley Publishing Company, 1987.

[20] E. Kreyszig, *Advanced Engineering Mathematics.* New York, NY: John Wiley & Sons, 5 ed., 1983.

[21] G. Daryanani, *Network Synthesis And Design.* New York, NY: John Wiley & Sons, 1976.

[22] A. S. Sedra and P. O. Bracket, *Filter Theory and Design: Active and Passive.* Beaverton, OR: Matrix Publishers, Inc., 1978.

[23] W. T. Sagun, F. H. Ives, G. L. Baldwin, and T. Hornak, "A 125-MHz 12-bit digital-to-analog converter system," *Hewlett-Packard Journal*, p. 78, April 1988.

[24] R. Hassun and A. W. Kovalick, "An arbitrary waveform synthesizer for DC to 50MHz," *Hewlett-Packard Journal*, pp. 69–77, April 1988.

[25] P. J. Lim and B. A. Wooley, "A high-speed sample-and-hold technique using a Miller hold capacitance," *JSSC*, vol. 26, pp. 643–651, April 1991.

[26] K. R. Stafford, P. R. Gray, and R. A. Blanchard, "A complete monolithic sample/hold amplifier," *JSSC*, vol. sc-9, pp. 381–387, December 1974.

[27] T. Miki, Y. Nakamura, M. Nakaya, S. Asai, Y. Akasaka, and Y. Horiba, "An 80-MHz 8-bit CMOS D/A converter," *JSSC*, vol. 21, pp. 983–988, December 1986.

[28] K. Nojima and Y. Gendai, "An 8-bit 800-MHz DAC," *JSSC*, vol. 25, pp. 1353–1359, December 1990.

[29] M. J. M. Pelgrom, "A 10-b 50-MHz CMOS D/A converter with 75-Ω buffer," *JSSC*, vol. VOL. 25, pp. 1347–1352, December 1990.

[30] T. Kamoto, Y. Akazawa, and M. Shinagawa, "An 8-bit 2-ns monolithic DAC," *JSSC*, vol. 23, pp. 142–146, February 1988.

[31] K. Maio, S.-I. Hayashi, M. Hotta, T. Watanabe, S. Ueda, and N. Yokozawa, "A 500-MHz 8-bit D/A converter," *JSSC*, vol. 20, pp. 1133–1136, December 1985.

[32] *AD568 DAC Data Sheet.* Analog Devices, Inc., 1992.

[33] U. Cilingiroglu, *Systematic Analysis of Bipolar and MOS Transistors.* Boston, MA: Artech House, 1993.

[34] D. A. Hodges and H. G. Jackson, *Analysis and Design of Digital Integrated Circuits.* New York, NY: McGraw-Hill Book Company, 1983.

[35] R. B. Craven, "An integrated circuit 12-bit D/A converter," *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, February 1975.

[36] D. J. Dooley, "A complete monolithic 10-b DAC," *IEEE Journal of Solid-State Circuits*, December 1973.

[37] A. Cremonesi, F. Maloberti, and G. Polito, "A 100-MHz CMOS DAC for video-graphic systems," *JSSC*, vol. 24, p. 635, June 1989.

[38] Y. Nakamura, T. Miki, A. Maeda, H. Kohdoh, and N. Yazawa, "A 10-b 70-MS/s CMOS D/A converter," *JSSC*, vol. 26, pp. 637–642, April 1991.

[39] J. J. Pastoriza, H. Krabbe, and A. A. Molinari, "A high performance monolithic d/a converter circuit," *ISSCC Digest of Technical Papers*, pp. 122–124, February 1970.

[40] R. J. V. de Plassche, "Dynamic element matching for high-accuracy monolithic D/A converters," *IEEE Journal of Solid-State Circuits*, December 1976.

[41] J. L. McCreary, "Matching properties, and voltage and temperature dependence of MOS capacitors," *IEEE Journal of Solid-State Circuits*, December 1981.

[42] J.-B. Shyu, G. C. Temes, and F. Krummenacher, "Random error effects in matched MOS capacitors and current sources," *IEEE Journal of Solid-State Circuits*, December 1984.

[43] K. R. Lakshmikumar, R. A. Hadaway, and M. A. Copeland, "Characterization and modeling of mismatch in MOS transistors for precision analog design," *IEEE Journal of Solid-State Circuits*, December 1986.

[44] G. Kelson, H. H. Stellrecht, and D. S. Perloff, "A monolithic 10-b digital-to-analog converter, using ion implantation," *IEEE Journal of Solid-State Circuits*, December 1973.

[45] D. T. Comer, "A monolithic 12-bit DAC," *IEEE Transactions on Circuits and Systems*, July 1978.

[46] P. Holloway and M. Norton, "A high yield, second generation 10-bit monolithic DAC," *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, February 1976.

[47] G. Erdi, "A precision trim technique for monolithic analog circuits," *IEEE Journal of Solid-State Circuits*, December 1975.

[48] J. J. Price, "A passive laser-trimming technique to improve the linearity of a 10-bit D/A converter," *IEEE Journal of Solid-State Circuits*, December 1976.

[49] H.-S. Lee, D. A. Hodges, and P. R. Gray, "A self-calibrating 15 bit CMOS A/D converter," *IEEE Journal of Solid-State Circuits*, December 1984.

[50] D. W. Groeneveld, H. J. Schouwenaars, H. A. H. Termeer, and C. A. A. Bastiaansen, "A self-calibration technique for monolithic high-resolution D/A converters," *IEEE Journal of Solid-State Circuits*, vol. 24, pp. 1517–1522, December 1989.

[51] D. A. Kerth, N. S. Sooch, and E. J. Swanson, "A 12-bit 1-MHz two-step flash ADC," *IEEE Journal of Solid-State Circuits*, vol. 24, pp. 250–5, April 1989.

[52] D. A. Mercer, "Two approaches to increasing spurious free dynamic range in high speed dacs," *Proceedings of the Bipolar Circuits Technology Meeting*, October 1993.

[53] *AD9712/3 100MSPS D/A Converter Data Sheet.* Analog Devices, Inc., 1990.

[54] *DAC600 12-bit 256MHz DAC Data Sheet.* Burr-Brown, Inc., 1991.

[55] M. Nayebi and B. A. Wooley, "A 10-bit video BiCMOS track-and-hold amplifier," *JSSC*, vol. 24, pp. 1507–1516, December 1989.

[56] K.-C. Hsieh, T. A. Knotts, G. L. Baldwin, and T. Hornak, "A 12-bit 1-Gword/s GaAs digital-to-analog converter system," *JSSC*, vol. 22, pp. 1048–1054, December 1987.

[57] F. H. Irons, D. M. Hummels, and S. P. Kennedy, "Improved compensation for analog-to-digital converters," *IEEE Transactions on Circuits and Systems*, vol. 38, pp. 958–961, August 1991.

[58] H. Samueli, "The design of multiplierless FIR filters for compensating D/A converter frequency response distortion," *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 1064–1066, August 1988.

[59] T. Lin and H. Samueli, "A 200-MHz CMOS x/sin(x)digital filter for compensating D/A converter frequency response distortion," *JSSC*, vol. 26, pp. 1278–1285, September 1991.

[60] G. Lowitz and R. Armitano, "'Predistortion' improves digital synthesizer accuracy," *Electronic Design*, pp. 85–89, March 31, 1988.

[61] G. R. Spalding and R. L. Geiger, "Digital correction for improved spectral response in signal generation systems," *Proceedings of the international symposium on circuits and systems*, May 1993.

[62] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proceedings of the IEEE*, vol. 66, pp. 51–84, January 1978.

[63] E. G. Soenen, *Error Modeling, Self-Calibration and Design of Pipelined Analog to Digital Converters.* PhD Dissertation: Texas A&M University, May 1992.

APPENDIX A. PRE-DISTORTION SIMULATION PROGRAM

```
/*
This program is used to simulate the predistortion algorithm.  It
includes subroutines that model the DAC and its non-idealities
(INL, DNL, Cap coupling, non-linear time constants...) as well
as an FFT routine for detecting and pre-distorting for these non-
idealities.  The FFT is very similiar to that in the notebook, except
that it returns phase.

The program outputs to 2 specified files, one for spectral data on
the dac, and one for the time domain dac output, both are updated
each iteration.  The current format for invocation is:

        d spec time > cor

where cor gets the mag and phase of the corrections at each itteration.

Version #6 has a new algorithm which corrects all spectral
errors above a threshold instead of pushing down on the unwanted
components (this algorithm will correct the desired spectral compenents
as well as the undesired).

Version #7 has transient response in the DAC (and therefore the
dac output fft has more points than the digital input fft).

Version #8 has quantization and range limitations.  Signals that
go outside the range do so because of INL and DNL errors.  If the
digital signal is out of range however, the signal will be limited
to the largest possible code.  This means that there must be extra
headroom in the dac system to correct "full-scale" signal.

Version #9 has been ported to vincent and the line in sample that
returns the number of points has been changed from:
return(npoints+1) to return(npoints)
It seems that this should have been done on HP's as well.

Version #10 moves all of the variables into the main body of the
program and reads them in from an external file to avoid frequent
recompiling.  The new execution line is:
                        d i s t > c
where d is the executable and i is the new input file.  If the original
method is desired, commenting out the readin call and the opening and
closing of the file fin will cause execution of the algorithm with
the default values contained in the program listing.

The routine disto has also been altered in this version to calculate the
SFDR at each iteration IF THE SIGNAL IS SYNCHRONOUS WITH THE CLOCK,
meaning that the FFT places all input spectral components in a single
bin rather then smearing them across many.  Windows may need to be
added to generalize the calculation.  The SFDR is printed out to
the screen (usually piped to the file c).

A filter routine has been added that computes the output of a simple,
two complex pole transfer function.

*/

#include <stdio.h>
#include <math.h>
#define ASIZE   5000
#define MAXLINE 100

main(argc,argv)
```

```
int argc;
char *argv[];

        {
/* System */
        double dig[ASIZE];              /* ideal digital input */
        double ism[ASIZE],isp[ASIZE];   /* ideal spectrum mag + phase */
        double sm[ASIZE],sp[ASIZE];     /* current mag, phase(cos) */
        double cor[ASIZE];              /* correction signal */
        double new[ASIZE];              /* newly corrected dig sig */
        double out[ASIZE];              /* dac output from new  */
        int i,count,flag;
        int m,n;                        /* # dig & analog points */

        int itlimit = 30;               /* iteration limit */
        int plimit = 60;                /* # of spectral bins printed    */
        int bw = 40;                    /* bandwidth of interest in bins */
        double bound = 100.0e-6;        /* correction threshold */

/* DAC  */
        int over = 2;                   /* # FFT points per sample period */
        double  dnl = 0.01;             /* DNL max value */
        int    ds = 1;                  /* DNL sign (overlap?)   */
        double dc = 0.010;              /* The DC offset   */
        double i1 = 0.010;              /* The gain error  */
        double i2 = 0.020;              /* INL coefficient */
        double i3 =  0.020;             /* INL coefficient */
        double i4 = 0.000;              /* INL coefficient */
        double i5 = 0.01;               /* INL coefficient */
        int nbits = 13;                 /* number of bits in dac */
        double range = 1.5;             /* pos & neg dac boundary */

/* FILTER */
        double a = 10.0e6;      /* related to Q  */
        double b = 100.0e12;    /* Wo squared    */

/* Input Sampling */
        double  fs = 128.0e6;   /* sampling frequency           */
        double  time = 1e-6;    /* total time of simulation     */
        double  a1 = 0.50;      /* amplitude of signal          */
        double  f1 = 3.00e6;    /* freq of signal               */
        double  phi1=0.0;       /* phase of signal in radians   */
        double  a2 = 0.25;
        double  f2 = 7.0e6;
        double  phi2 = 2.356;
        double  a3 = 0.125;
        double  f3 = 12.0e6;
        double  phi3 = 0.61087;

/* Files */
        FILE *fin,*fopen();     /* input file (system & DAC variables) */
        FILE *fout1,*fopen();   /* spectrum file */
        FILE *fout2,*fopen();   /* time signal file (slows exec) */


        fin = fopen(*++argv,"r");
/*
        printf("%le %le %le %le %le %le %le \n\n",bound,dnl,dc,i2,i3,i4,i5);
        printf("%le %le %le %le %le %le %le \n\n",range,fs,time,a1,f1,phi1,a2);
        printf("%le %le %le %le %le %le %le \n\n",f2,phi2,a3,f3,phi3,a,b);
        printf("%d %d %d %d %d %d \n\n",itlimit,plimit,bw,over,ds,nbits);
*/
```

```
        readin(fin,&itlimit,&plimit,&bw,&bound,
                &over,&dnl,&ds,&dc,&i1,&i2,&i3,&i4,&i5,&nbits,&range,
                &fs,&time,&a1,&f1,&phi1,&a2,&f2,&phi2,&a3,&f3,&phi3,&a,&b);
        fclose(fin);
/*
        printf("%le %le %le %le %le %le %le \n\n",bound,dnl,dc,i2,i3,i4,i5);
        printf("%le %le %le %le %le %le %le \n\n",range,fs,time,a1,f1,phi1,a2);
        printf("%le %le %le %le %le %le %le \n\n",f2,phi2,a3,f3,phi3,a,b);
        printf("%d %d %d %d %d %d \n\n",itlimit,plimit,bw,over,ds,nbits);
*/
        fout1 = fopen(*++argv,"w");
        fout2 = fopen(*++argv,"w");
        m = sample(dig,fs,time,a1,f1,phi1,a2,f2,phi2,a3,f3,phi3);
        fft(dig,ism,isp,m);
        fprintf(fout1,"#The ideal spectrum (initial): \n#\n");
        fprintf(fout1,"#freq        mag                phase \n");
        for (i = 0; i <= plimit; i++) {
                fprintf(fout1,"#%d      %f      %f \n",i,ism[i],isp[i]);
                }
        for (i = 0; i < m; i++) {
                new[i] = dig[i];
                cor[i] = 0.0;
                }
        fprintf(fout2,"#The digital input signal: \n#\n");
        for (i=0;i<m;i++) {
                fprintf(fout2,"#%d %f \n",i,dig[i]);
                }
        flag = 1;
        count = 0;
        while (flag && (count <= itlimit)) {
                count++;
                for (i = 0; i < m; i++) {
                        new[i] += cor[i];
                        }
                n=m*over;
                dac(new,out,m,n,over,nbits,range,dnl,ds,dc,i1,i2,i3,i4,i5);
                fprintf(fout2,"#The dac output (%d): \n\n",count);
                for (i=0;i<n;i++) {
                        fprintf(fout2,"%d %f \n",i,out[i]);
                        }
/*
                fprintf(fout2,"#The dac error signal: \n\n");
                for (i=0;i<n;i++) {
                   fprintf(fout2,"%f %f \n",3.0*i/m-1.5,out[i]+1.5-3.0*i/m);
                        }
                filter(out,a,b,time,n);
*/
                fprintf(fout2,"#The filter output (%d): \n\n",count);
                for (i=0;i<n;i++) {
                        fprintf(fout2,"%d %f \n",i,out[i]);
                        }
                fft(out,sm,sp,n);
                fprintf(fout1,"#New (%d) spectrum: \n\n",count);
                for (i = 0; i <= plimit; i++) {
                        fprintf(fout1,"%d      %f %f \n",i,sm[i],sp[i]);
                        }
                printf("#Correction #%d \n",count);
                flag = disto(ism,isp,sm,sp,cor,m,bw,bound);
                }
        fclose(fout1);
        fclose(fout2);
        return(0);
        }
```

```
/* The READIN routine:
                This is VERY file sensitive, and not robust!  */

readin(fin,itlimit,plimit,bw,bound,over,dnl,ds,dc,i1,i2,i3,i4,i5,nbits,
                range,fs,time,a1,f1,phi1,a2,f2,phi2,a3,f3,phi3,a,b)

FILE    *fin;
double  *bound,*dnl,*dc,*i1,*i2,*i3,*i4,*i5,*range;
double  *fs,*time,*a1,*f1,*phi1,*a2,*f2,*phi2,*a3,*f3,*phi3,*a,*b;
int     *itlimit,*plimit,*bw,*over,*ds,*nbits;

        {
        char    line[MAXLINE];

        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%d",itlimit);
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%d",plimit);
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%d",bw);
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%le",bound);
        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%d",over);
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%le %d",dnl,ds);
        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%le %le %le %le %le %le",dc,i1,i2,i3,i4,i5);
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%d",nbits );
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%le",range);
        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%le",fs);
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%le",time);
        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%le %le %le",a1,f1,phi1);
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%le %le %le",a2,f2,phi2);
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%le %le %le",a3,f3,phi3);
        fgets(line,MAXLINE,fin);
        fgets(line,MAXLINE,fin);
        fscanf(fin,"%le %le",a,b);
        fgets(line,MAXLINE,fin);
```

```
        }

/* Sets up digital input signal
        For the signal to be coherent with the clock:  time*f1=integer
                where the integer is the eventual bin location
        For the FFT to work:  time*fs=binary                    */

sample(dig,fs,time,a1,f1,phi1,a2,f2,phi2,a3,f3,phi3)
double  dig[];
double fs,time,a1,f1,phi1,a2,f2,phi2,a3,f3,phi3;

        {
        int i;
        double  t;
        int     npoints;

        npoints = time*fs;      /* notice truncation */
        printf("#npoints = %d \n",npoints);
        /* Define the ideal digital input signal */
        for (i = 0; i <= npoints; i++) {
                t = i/fs;
                dig[i] = 0.0;
                /* Ramp */
/*
                dig[i] += (4.0*i/(npoints*1.0)) -2.00;
*/
                /* sinusiods of input spectrum  */
                dig[i] += a1*cos(2*3.141592654*f1*t + phi1);
                dig[i] += a2*cos(2*3.141592654*f2*t + phi2);
                dig[i] += a3*cos(2*3.141592654*f3*t + phi3);
/*
                dig[i] += a1*cos(2*3.141592654*f1*t + phi1*(i/64));
*/
                }
        return(npoints);
        }

dac(new,out,m,n,over,nbits,range,dnl,ds,dc,i1,i2,i3,i4,i5)
double  new[],out[],range,dnl,dc,i1,i2,i3,i4,i5;
int     m,n,over,nbits,ds;
        {
        int i,j,lim,code;
        double  temp[ASIZE],newq[ASIZE],q;

        /* Distort final settled value of dac output      */
        lim = pow(2.0,nbits*1.0);
        q = 2*range/lim*1.0;
        for (i = 0; i < m; i++) {
                /* Quantization and Range Limitation */
                code = (new[i]+range)/q;        /* truncation */
                if (code > lim -1)
                        code = lim -1;
                if (code < 0)
                        code = 0;
                newq[i] = code*q - range;
                /* DNL distortion:              */
                if (newq[i]>0) {
                        if (newq[i]>0.5*range)
                          temp[i] = newq[i]*(1+ds*dnl/2) - ds*range*dnl/2;
                        else
                          temp[i] = newq[i]*(1+ds*3*dnl/2) - ds*range*dnl/2;
                        }
                else    {
```

```
                        if (newq[i]>-0.5*range)
                          temp[i] = newq[i]*(1+ds*3*dnl/2) + ds*range*dnl/2;
                        else
                          temp[i] = newq[i]*(1+ds*dnl/2) + ds*range*dnl/2;
                        }

                /* INL distortion:                                      */
                temp[i] *= i1;
                temp[i] += dc;
                temp[i] += i2*newq[i]*newq[i];
                temp[i] += i3*newq[i]*newq[i]*newq[i];
                temp[i] += i4*newq[i]*newq[i]*newq[i]*newq[i];
                temp[i] += i5*newq[i]*newq[i]*newq[i]*newq[i]*newq[i];
                }
        /* This portion of the dac expands the output points to include
        transient behavior.                                             */

        n = over*m;
        for (i = 0; i < m; i++) {
                for (j = 0; j < over; j++) {
                        out[i*over+j] = temp[i];
/*
                        out[i*over+j] += 0.4*temp[i]*temp[i]*(1.0-j*1.0/over)
                                                *(1.0-j*1.0/over);
                        out[i*over+j] += (1.0-j*j*1.0/(over*over*1.0))*temp[i];
*/
                        }
                }
        }

/*  This routine filters with a forward Euler representation of
        H(s) = b/(s*s + as + b)    */

filter(out,a,b,time,n)
double out[],a,b,time;
int n;

        {
        double y1,y2;           /* y(n-1),y(n-2)        */
        double y[ASIZE];        /* temp storage         */
        double k;
        int i;

        a = a*time/(n*1.0);
        b = b*time*time/(n*n*1.0);
        k = 1/(1 + a + b);
        /* printf("# %le   %le \n",a,b); */
        y1 = 0.0;
        y2 = 0.0;
        for (i = 0; i < n; i++) {
                y[i] = k*(b*out[i] + (2+a)*y1 - y2);
                y2 = y1;
                y1 = y[i];
                }
        y1 = y[n-1];
        y2 = y[n-2];
        for (i = 0; i < n; i++) {
                y[i] = k*(b*out[i] + (2+a)*y1 - y2);
                y2 = y1;
                y1 = y[i];
                }
        for (i = 0; i < n; i++) {
                out[i] = y[i];
```

```
                }
        }

/*  This routine calculates the necessary correction needed in the
        digital signal based on the ideal and most recent spectrums.
        The variables worst and sfdr are the bin # of the current
        worst distortion values.  sfdr has no desired signal in that
        bin, and worst makes no distinction (so they are usually the
        same).*/

disto(ism,isp,sm,sp,cor,m,bw,bound)
double ism[],isp[],sm[],sp[],cor[],bound;
int    m,bw;

        {
        int i,j,flag,worst,sfdr;
        double a,b,arg,cr,ci,cm,cp;
        double w,s;

        flag = 0;
        for (j = 0; j < m; ++j)
                cor[j] = 0.0;
        worst = 999;
        sfdr  = 999;
        w = 0.0;
        s = 0.0;
        for (i = 0; i < bw; ++i) {     /* the bandwidth threshold (in bins) */
                a = pow(10.0,ism[i]/20);
                b = pow(10.0,sm[i]/20);
                cr = a*cos(isp[i]) - b*cos(sp[i]);
                ci = a*sin(isp[i]) - b*sin(sp[i]);
                cm = sqrt(cr*cr + ci*ci);
                if (cm > w) {
                        w = cm;
                        worst = i;
                        }
                if ((pow(10.0,ism[i]/20) < bound) && (cm > s)) {
                        sfdr = i;
                        s = cm;
                        }
                if ((ci == 0.0) && (cr == 0.0))
                        cp = 0.0;
                else
                        cp = atan2(ci,cr);
                if (cm > bound) {
                        flag = 1;
                        printf("%d %le %le \n",i,20*log10(cm),cp);
                        for (j = 0; j < m; ++j) {
                                arg = 2*3.141592654*i*j/(1.0*m);
                                cor[j] += cm*cos(arg + cp);
                                }
                        }
                }
        printf("#Worst error: %le in bin #%d \n",-20*log10(w),worst);
        printf("#SFDR:        %le in bin #%d \n",-20*log10(s),sfdr);
        return(flag);
        }

fft(dig,sm,sp,n)
double  dig[],sm[],sp[];
int     n;
        {
        double wr[ASIZE],wi[ASIZE];
```

```
        double spectrumr[ASIZE];
        double spectrumi[ASIZE];

        if (binary(n)) {
                init_w(n,wr,wi);
                fmult(n,wr,dig,spectrumr);
                fmult(n,wi,dig,spectrumi);
                mag(n,spectrumr,spectrumi,sm);
                phase(n,spectrumr,spectrumi,sp);
                }
        else
                printf("Error:  n = %d \n",n);
        return(0);
        }

binary(n)
int n;
        {
        int i = 0;

        while ((n > 2) && ((n % 2) == 0)) {
                n /= 2;
                i++;
                }
        if (n == 2)
                return(++i);
        else
                return(0);
        }

init_w(n,wr,wi)
int    n;
double wr[],wi[];
        {
        int i,j;
        double theta;

        for (i = 0; i <= n/4; ++i) {
                theta = 3.141592654*2*i/n;
                wi[i] = sin(theta);
                wr[i] = cos(theta);
                }
        for (i = 0; i <= n/4; ++i) {
                wi[n/2-i] = wi[i];
                wr[n/2-i] = wi[((3*n)/4)-i] = 0.0-wr[i];
                wr[n/2+i] = -wr[i];
                if (i!=0) {
                        wr[n-i] = wr[i];
                        wi[n-i] = -wi[i];
                        }
                }
        return(0);
        }

fmult(n,a,b,z)          /* NOT a general matrix mult function */
int    n;               /* Only works for the FFT application */
double a[],b[],z[];

        {
        int i,j;

        for (i = 0; i < n; ++i) {
                z[i] = 0;
```

```
                for (j = 0; j < n; ++j) {
                        z[i] += a[(j*i)%n] * b[j];
                        }
                }
        return(0);
        }

mag(n,a,b,z)
int    n;
double a[],b[],z[];

        {
        int i;
        for (i = 0; i < n; ++i) {
                z[i] = a[i]*a[i] + b[i]*b[i];
                z[i] = sqrt(z[i])*2/n;
                if (i == 0)
                        z[0] = z[0]/2;    /* Special DC scale factor */
                if (z[i] < 1.0e-12)       /* avoid log10(0)          */
                        z[i] = -180;
                else
                        z[i] = 20 * log10(z[i]);
                }
        }

phase(n,a,b,z)
int     n;
double  a[],b[],z[];

        {
        int i;
        for (i = 0; i < n; ++i)
                if ((b[i] == 0) && (a[i] == 0))
                        z[i] = 0.0;
                else
                        z[i] = atan2(-b[i],a[i]);
        }
```